USER GUIDE

&

SOFTWARE DOCUMENTATION

Cryogenic Goniometer
Motion Control Software

Version 1.0 - November 2004

Markus P. Hehlen
LANSCE-12, MS H805
Los Alamos National Laboratory
Los Alamos, NM 87545

LANL Software Category 3
TSPA ("Technology and Software Publicly Available")

Table of Contents

1. Purpose

This document describes the operating procedures, functionality, and structure of the LabVIEW-based software interface `GoniometerMotionControl.vi` controlling the Three-Axis Cryogenic Goniometer that has been designed and built by the B. Fultz group at Caltech in collaboration with Robert Chave Applied Physics Inc.

In the future, the cryogenic goniometer will be used on the ARCS neutron instrument at SNS. However it is currently located at the Pharos inelastic neutron spectrometer at the Lujan Neutron Scattering Center at Los Alamos National Laboratory for commissioning activities including finalizing the hardware, testing the performance, and conducting first single-crystal experiments as part of the Pharos user program.

Section 2 presents a User Guide that contains important safety information and gives a step by step description of the various screens of the motion control interface. This section can be used as a standalone text for the user who is not interested in the technical details of the motion control system. Sections 3 and 4 give a comprehensive description of the hardware and software, respectively. These sections contain important information relevant to understanding the concepts and implementation details of the motion control system, and they are relevant for any subsequent adaptation of the software to another neutron instrument. Finally, the Appendix contains the complete source code listings of all software pieces making up the goniometer motion control system.

The software interface developed at the Lujan Center contains both generic and Pharos-specific components. The interface was developed with the specific purpose of enabling the functionality needed to conduct single-crystal experiments on Pharos. Wherever possible, sub-routines were written such that they can be used in LabVIEW-based interfaces on other instruments. However, the software described here is intended to run on Pharos only, and certain instrument-specific modifications will be required to port the software to another neutron instrument. The well commented code in combination with this document should hopefully facilitate this task.

2. User Manual

2.1. Getting Started

2.1.1. Important Safety Precautions

This is a computer-controlled motion system. While the code was written and documented for maximum user-friendliness, safety, and reliability, there is a non-zero but negligible risk for minor injury or equipment damage by the moving parts of the goniometer if the goniometer is not installed in the Pharos sample well. This risk is zero once the goniometer assembly is installed in the Pharos sample well.

The operator(s) shall exercise caution when the goniometer is outside the Pharos sample well and motions are being executed with the goniometer head exposed. It is this work condition that requires particular attention and awareness by all operators involved. While it is unlikely that a single operator issuing a motion command from the PC exposes himself to the risks of a moving part on the goniometer located at some distance, special attention is needed when two or more workers are present. It is the responsibility of the operator at the PC to communicate with all co-workers in the area before initiating any goniometer motion. Remember: stepper motors are powerful and the stages use a high gear ratio. Therefore the force produced by the moving stages is large. These risks are adequately mitigated by adhering to the work procedures outlined in the following.

Never touch any moving parts of the goniometer. Stay clear of the space defined by the four legs of the goniometer stand whenever the goniometer control unit is powered up.

Before you work on the exposed goniometer head (like mounting samples or doing maintenance), perform all of the following steps:

1. power down the goniometer main unit via its power switch on the back panel
2. exit the LabVIEW-based `GoniometerMotionControl.vi` described in this document
3. exit the Parker Motion Planner application.

Once this state is reached, no motion can occur even if one of the three above elements were to be inadvertently activated. After completion of the work on the exposed goniometer head and before returning the system to the motion-ready state, perform all of the following steps:

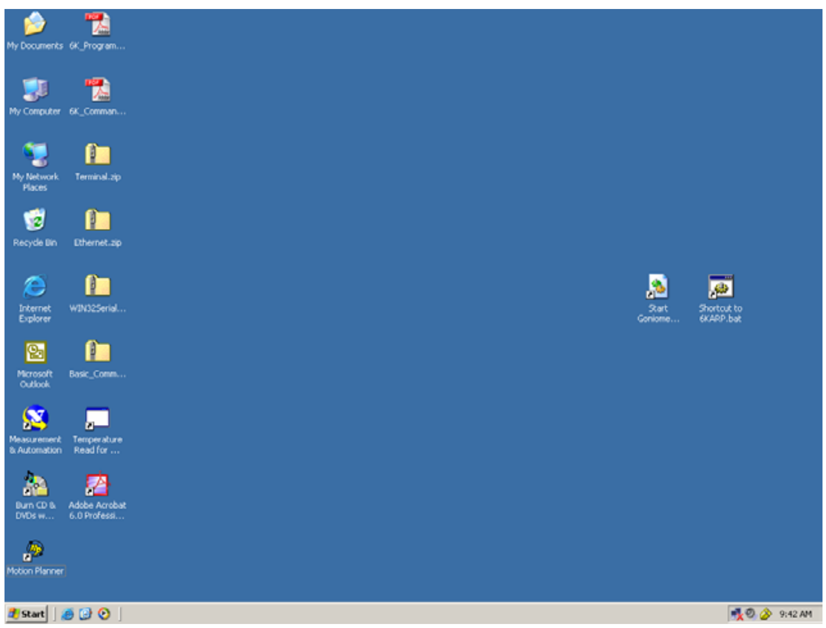
1. check the goniometer head area for any remaining foreign objects (tools, screws, nuts, foils, etc.) and to remove them
2. stand clear of the space defined by the four legs of the goniometer stand.

Only after completing these steps can the goniometer control unit be powered up and the software application be launched.

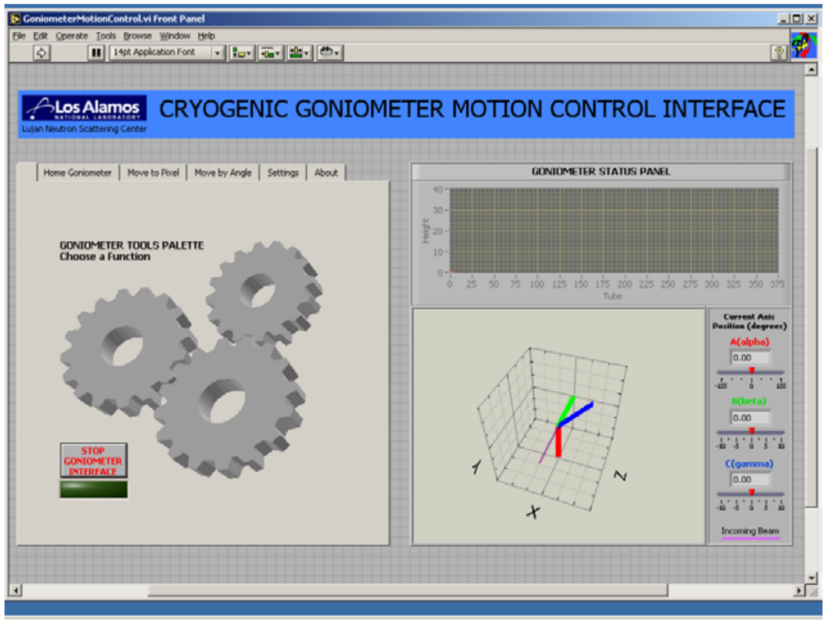
2.1.2. Motion System Startup

Follow the steps below to power up the goniometer and to launch the motion control application:

1. Check the goniometer head area for any remaining foreign objects (tools, screws, nuts, foils, etc.) and remove them. Also ensure that sample-related hardware does not interfere with the full range of motion of the three goniometer stages
2. Clear the space defined by the four legs of the goniometer stand
3. Power up the goniometer main unit via the power switch on the back panel
4. If the PC is OFF, power it up now; the system will automatically establish the Ethernet connection with the goniometer stepper motor controller.
   If the PC is already ON, double-click the "`Shortcut to 6KARP.bat`" on the desktop (shown below at middle right of desktop) to establish the Ethernet connection with the goniometer stepper motor controller.

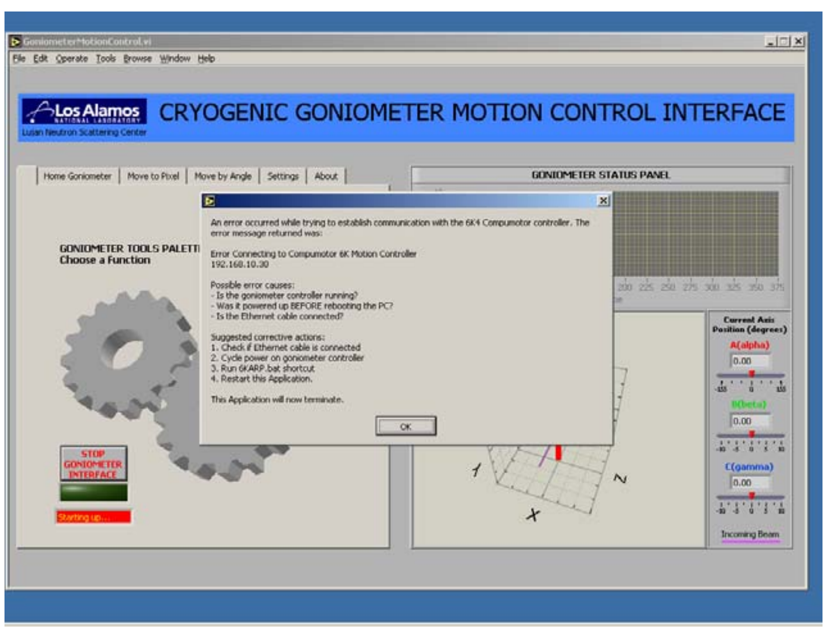


5. Double-click the "`Start Goniometer Motion Control.vi`" shortcut on the desktop. LabVIEW will be launched and the `GoniometerMotionControl.vi` will be opened.
6. Launch the `GoniometerMotionControl.vi` by clicking the "run arrow" icon  on the left side in the toolbar.



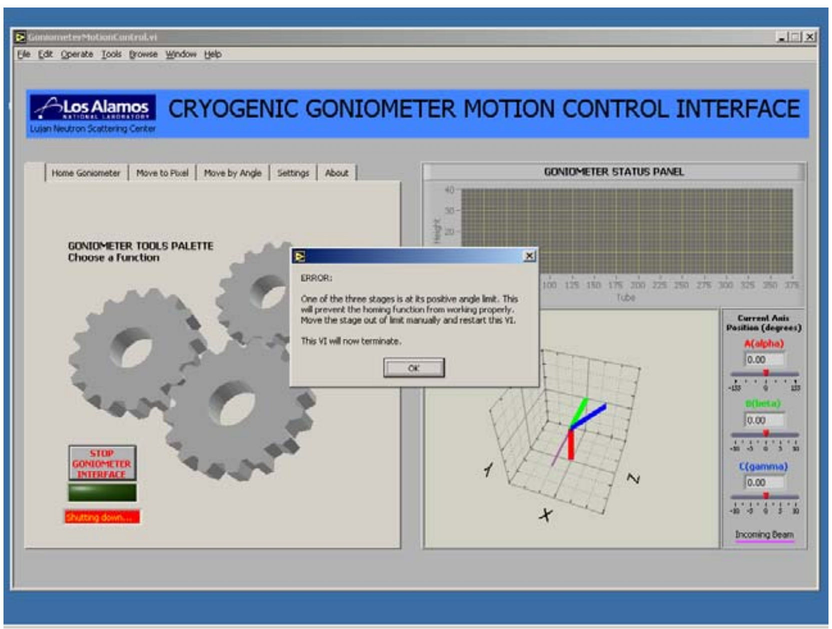
7. A blinking message indicates that the application is in the process of starting up. The user is presented with a dialog box reminding the operator to first execute the "Home Goniometer" function after starting up. Subsequently, several initialization tasks are being performed. If they execute successfully, the application presents the front panel and is ready for accepting commands as indicated by the green LED indicator. Proceed to Section 2.2.
8. Initialization Errors: three error conditions are possible during the initialization process, and they will all cause the `GoniometerMotionControl.vi` to terminate gracefully:

    a. The application cannot communicate with the 6K4 stepper motor controller.



There are several possible causes such as the goniometer controller unit not running, the 6KARP.BAT command having been issued after powering up the goniometer controller unit (either by booting the PC or by clicking the 6KARP.BAT shortcut on the desktop), or the Ethernet cable is not connected. Follow the instructions in the error dialog box to resolve the problem.

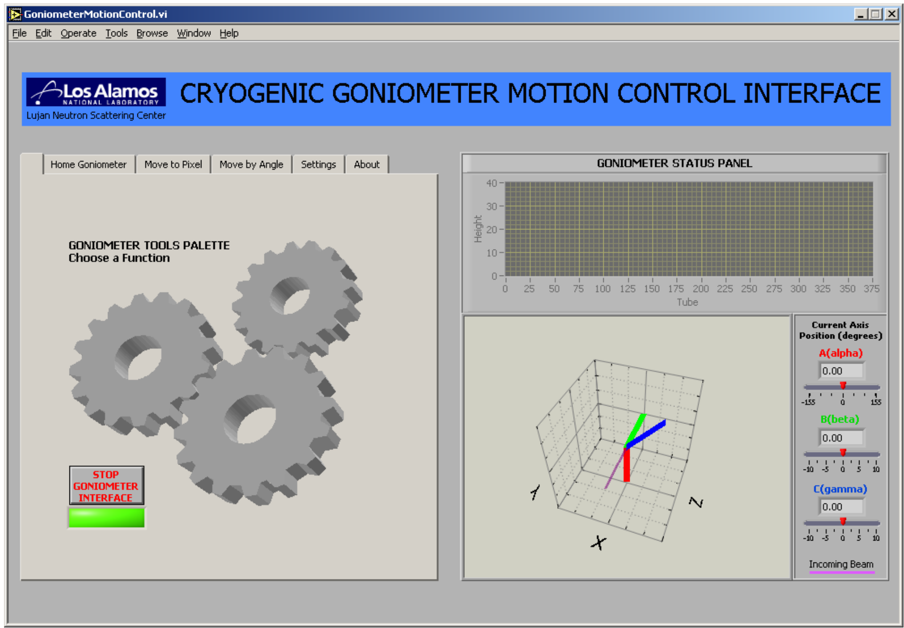
b. The application cannot communicate with the DAQ card which senses the stepper motor limit switches. This error indicates a configuration problem that has to be investigated using the National Instruments MAX application. All hardware is properly configured, and this error should not occur.
c. The application senses that one or more of the three goniometer stages is at its positive limit switch. Secure the system (see Section 2.1.1), manually move the stage(s) out of the limit condition, and restart the system and application.



2.2. Front Panel

The green indicator LED on the front panel indicates that the `GoniometerMotionControl.vi` is now running. The system is ready to accept commands and to perform motions.

The front panel is divided in three sections. The tabbed menu area on the left side is the Goniometer Tools Palette which enables the user to issue specific commands to the goniometer. The top right graph is a representation of the Pharos pixilated detector array and is used to visualize the location of single-crystal diffraction spots during operation. The lower right section visualizes the spatial orientation of the goniometer axes of rotation and gives numerical information on the current angular position of the three stages.



The goniometer has no absolute position encoders and therefore finds itself in an <u>undefined state</u> after system power-up and launching of the `GoniometerMotionControl.vi`.

The "Home Goniometer" function (see Section 2.3) must be called first to initialize the system and to move the stages to a well-defined home position.

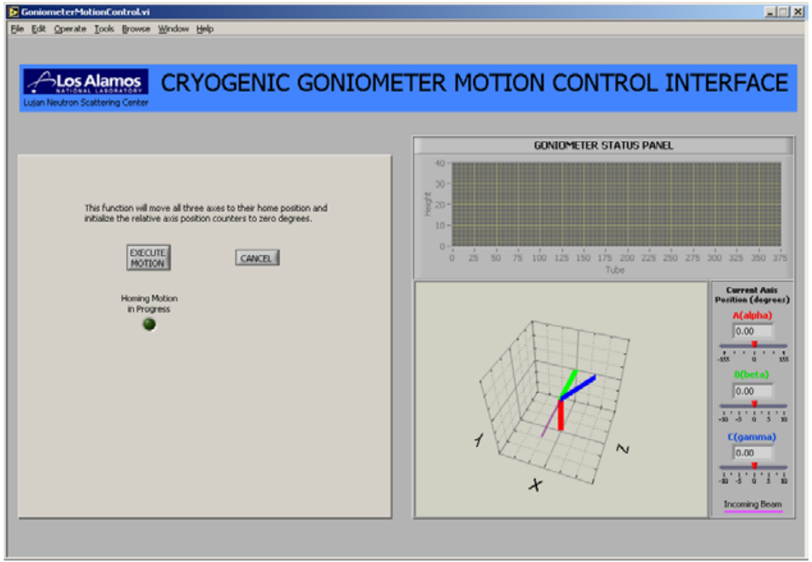
2.3. Home Goniometer

The "Home Goniometer" function initializes the motion system and moves each of the three stages to a well-defined "home" position from which all subsequent motions are tracked. This function can be called at any time during operation, but it must be called every time the `GoniometerMotionControl.vi` is launched.

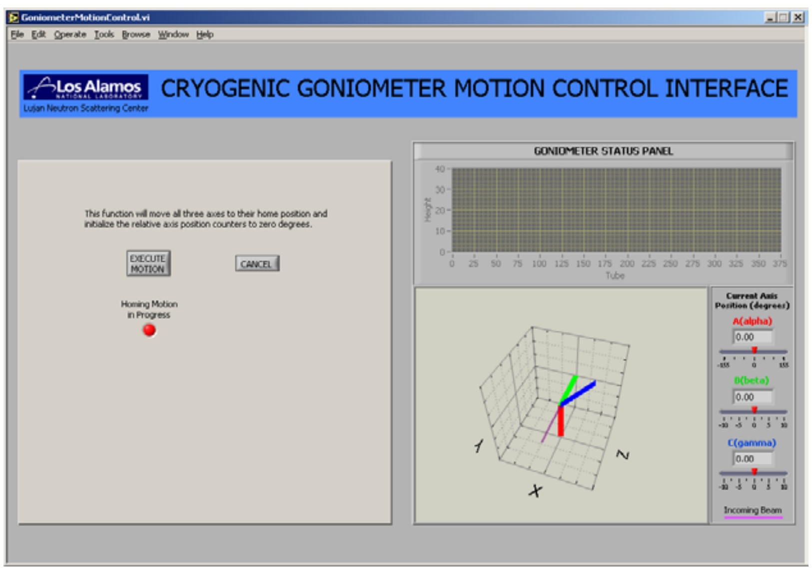
It is recommended that the "Home Goniometer" function is performed <u>after</u> the sample has been installed but <u>before</u> the goniometer is positioned in the Pharos sample well. Executing the homing function with the goniometer head still exposed serves as a visible readiness check of the motion control system before the goniometer is installed out-of-sight in the Pharos sample well.

To perform the "Home Goniometer" function follow these steps:

1. Click the "Home Goniometer" tab in the tools palette.



2. Click on the "Execute Motion" button to initiate the homing function, or click "Cancel" to exit this function without performing any motions.
3. If you execute the motion, a blinking LED will indicate that the motion is in progress.



4. Wait for the homing motion to complete and the `GoniometerMotionControl.vi` to return to the front panel.
5. The system is now initialized and ready to perform and track defined motions. The "homed" orientation of the three goniometer axes of rotation is visualized in the bottom right graph.

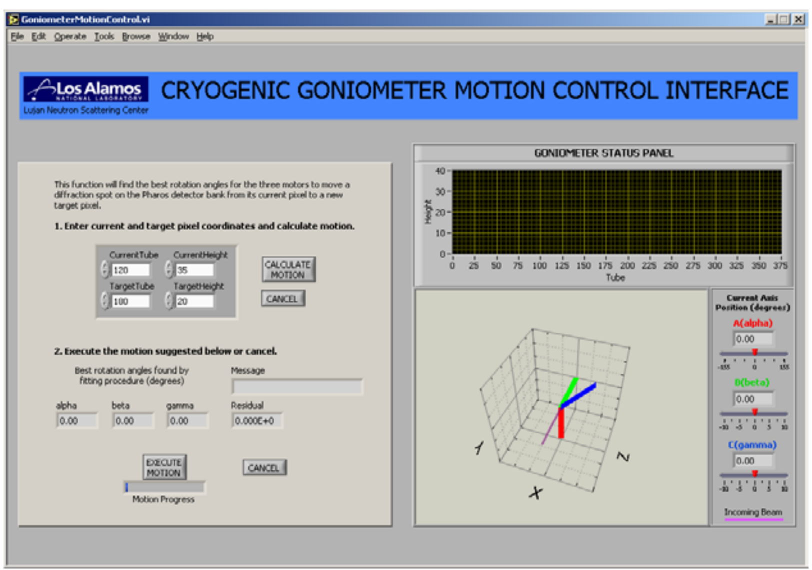
2.4. Moving to Pixel

The "Move to Pixel" function rotates the goniometer stages such that a diffraction spot located on the Pharos detector bank at a current pixel position is moved to a new target pixel position. Information on the current position is obtained by the user analyzing the data measured from a Pharos run, i.e. the motion control software has no connection to the DAQ system. Likewise, the motion control software has no knowledge of the crystal structure of the sample and thus the crystallographic assignment of the diffraction spots. The motion control software simply identifies and performs the 3-axis motion based on the goniometer spatial orientation and the user-supplied information on current and target orientation of a diffraction direction.

The "Move to Pixel" function is divided in two steps: first, the software calculates the required motions given a current and a target pixel location. The calculation ensures that all motions remain within the maximum ± angular range of each of the stages Second, the user initiates the suggested motion.

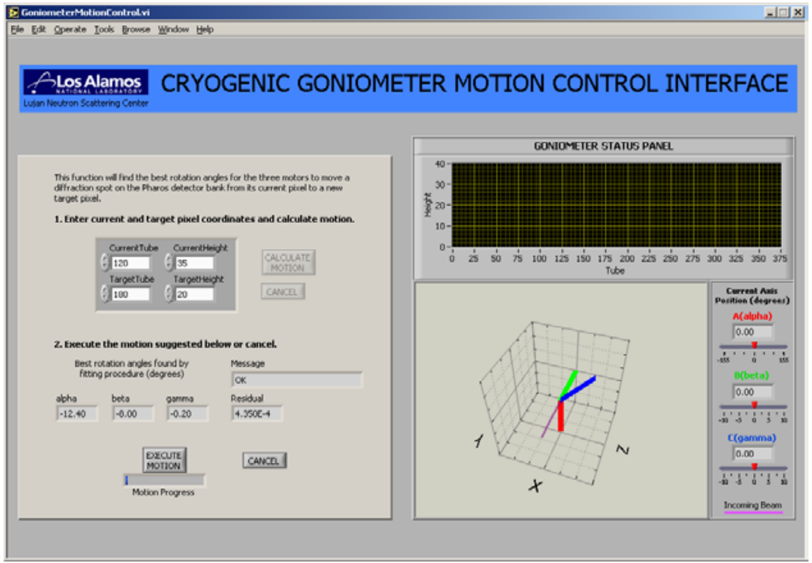
To perform the "Move to Pixel" function follow these steps:

1. Click the "Move to Pixel" tab in the tools palette.
2. Enter the current and target tube number and height pixel in the four respective numerical windows. The entries can but do not have to be terminated by pressing ENTER on the keyboard.

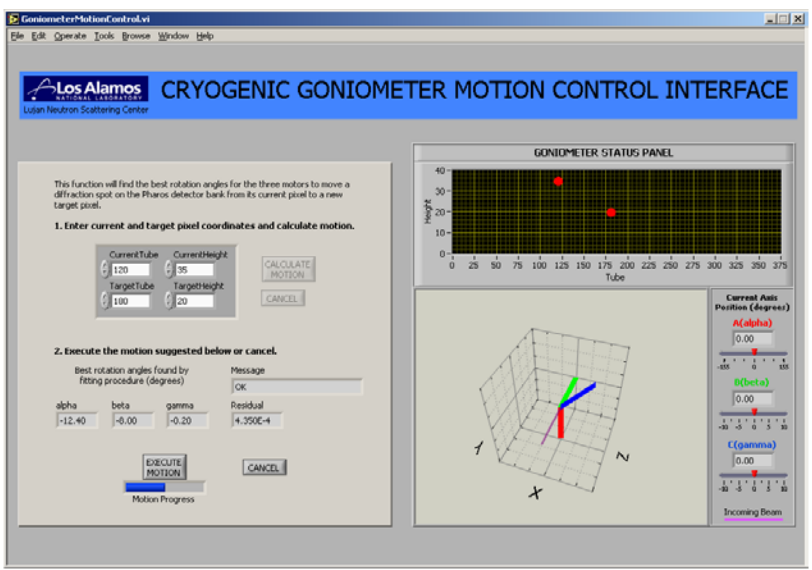


3. Click "Calculate Motion" to initiate the calculation of the best 3-axis motion, or click "Cancel" to exit the function without doing any calculations or motions.
4. After a few seconds, the three suggested rotation angles are returned in the numerical windows below, along with a residual and a message. The current and target diffraction spots are also visualized in the top right graph. The residual is a quantitative measure of the expected accuracy of the suggested motion, and the message interprets the residual value for the user. The message is "OK" if the suggested motion is expected to result in accurate positioning of the current

diffraction spot onto the requested target position. Otherwise, the message will indicate that inaccurate positioning is to be expected.



5. Click "Execute Motion" to initiate the suggested 3-axis motion, or click "Cancel" to exit this function without performing any motions.
6. Wait for the "Move to Pixel" motion to complete and the `GoniometerMotionControl.vi` to return to the front panel. The progress of the "Move to Pixel" motion is indicated by the progress bar.



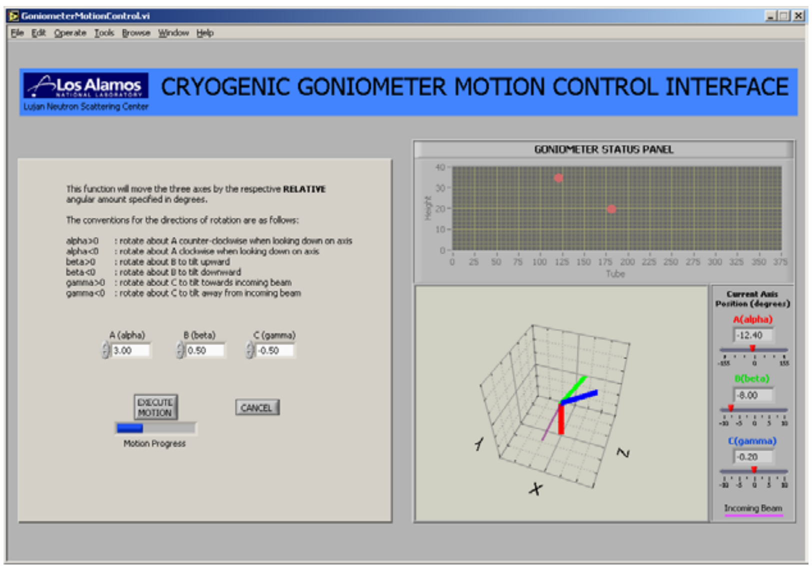
7. Back at the front panel, the new orientation of the three goniometer axes of rotation is visualized in the bottom right graph.

2.5. Moving by Angle

The "Move by Angle" function rotates the three stages by a respective user-defined angle. This is a general purpose function and can be used to perform sizeable reorientations of the stage or obvious angular corrections.

To perform the "Move by Angle" function follow these steps:

1. Click the "Move by Angle" tab in the tools palette.
2. Enter the angles in degrees for the individual stages to be rotated. The comments on the function page help you correctly choose the signs of the angles. One, two, or all stages will be moved by the specified angles at the same time. The software will not allow the user to perform motions that would drive any of the stages into a limit switch. Out-of-range requests on any of the three axes will simply be ignored by returning to the Front Panel.



3. Click "Execute Motion" to initiate the specified rotations, or click "Cancel" to exit this function without performing any motions.
4. Wait for the "Move by Angle" motion to complete and the `GoniometerMotionControl.vi` to return to the front panel. The progress of the "Move by Angle" motion is indicated by the progress bar.
5. Back at the front panel, the new orientation of the three goniometer axes of rotation is visualized in the bottom right graph.

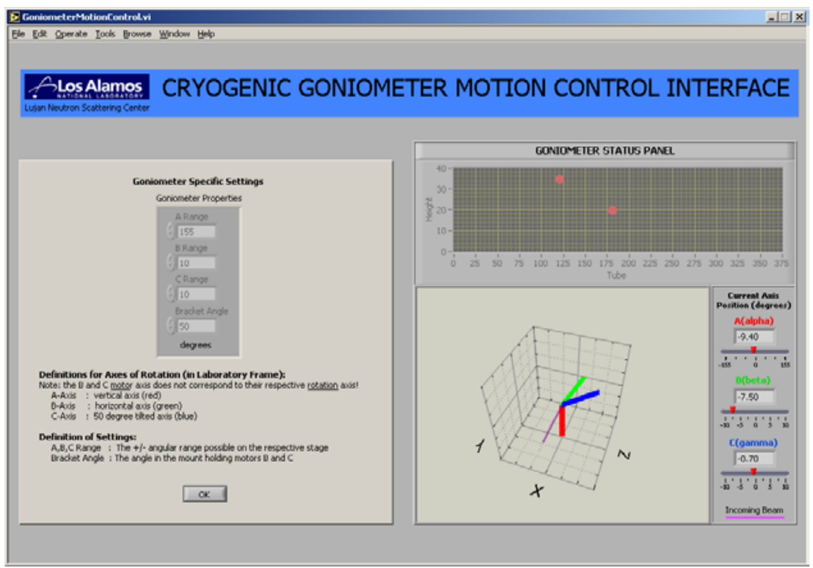
2.6. System Parameters

The “Settings” function displays goniometer-specific parameters. These parameters are preset for the goniometer hardware and are automatically loaded when starting the `GoniometerMotionControl.vi`.

There is no need to change any of these parameters from the values shown below. Changing the parameters can adversely affect the accuracy of all motions and can result in potentially undesired motions. Therefore, all controls are disabled.

To perform the “Settings” function follow these steps:

1. Click the “Settings” tab in the tools palette.



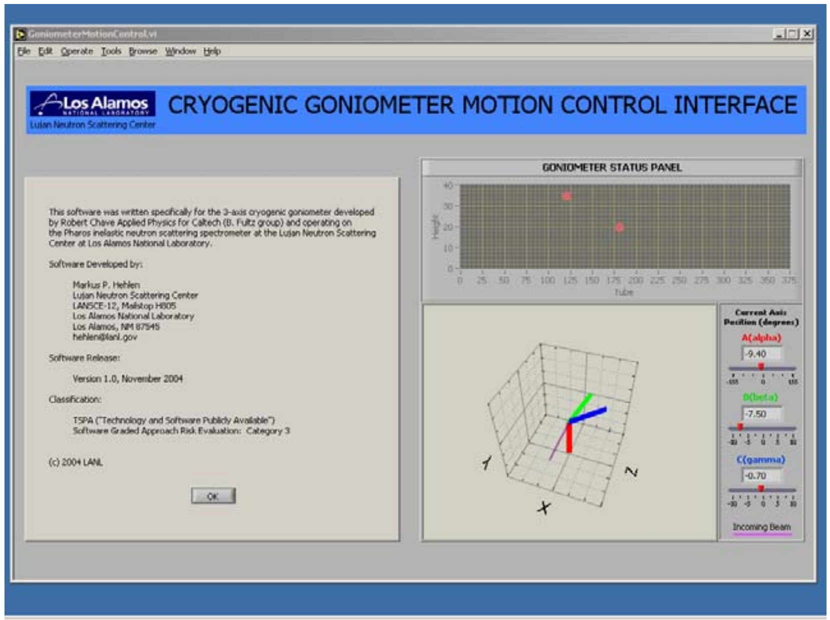
2. Click “OK” to return to the `GoniometerMotionControl.vi` front panel.

2.7. About `GoniometerMotionControl.vi`

The “About” page contains information on the software developer, the current version installed, and the LANL software classification.

To perform the “About” function follow these steps:

1. Click the “About” tab in the tools palette.



2. Click "OK" to exit this page and to return to the `GoniometerMotionControl.vi` front panel.

2.8. Exiting the `GoniometerMotionControl.vi`

The `GoniometerMotionControl.vi` application is exited by clicking the red "STOP GONIOMETER INTERFACE" button on the front panel in the tools palette. The application will shut down as indicated by the green LED indicator being off. Click File>Exit to exit labVIEW.

	Once you exit the `GoniometerMotionControl.vi` application, all information on the current goniometer orientation is lost and "homing" of the goniometer is necessary. Therefore, after starting up again, perform the "Home Goniometer" function as described in Section 2.3.

3. Hardware Documentation

3.1. Three-Axis Goniometer

The system hardware consists of four sub-systems as shown in Figure 1: (1) the goniometer assembly with its stand, (2) the compressor unit for the closed-cycle helium refrigerator, (3) the goniometer main unit, and (4) a dedicated personal computer. The focus here is solely on the aspects of motion control and not on the cryogenic performance and control.

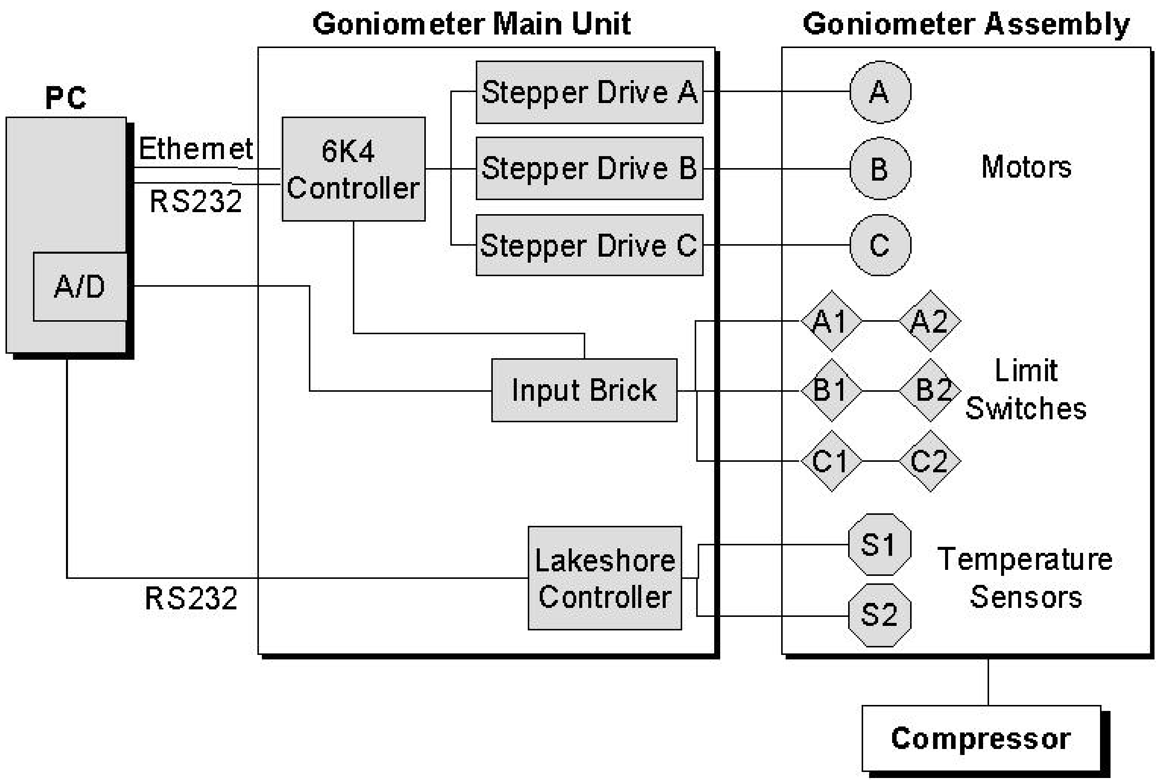


Figure 1: System Schematic of Three-Axis Cryogenic Goniometer Hardware

The goniometer assembly consists of a top plate that serves as both the vacuum flange for the sample area and the electrical interface. The Displex is mounted rigidly in the center of the top plate. A three-axis motion system can rotate a sample located axially 24¼" below the top plate bottom surface. The three axes are stacked: a first rotation stage **A** (angle α) enables rotation about the vertical axis in the laboratory frame of reference, rotating with it the rotation stages **B** (angle β) and **C** (angle γ) that are mounted to **A** on an arm. Rotation stage **B** provides rotation about an axis that is perpendicular to that of stage **A** and that is horizontal in the laboratory frame of reference. Stage **B** rotates with it stage **C** which is mounted to it. Stage **C** provides rotation about an axis that is tilted at an angle of 50° with respect to the laboratory frame vertical. The rotating sample is connected to the Displex cold-finger by means of litz wire.

The goniometer assembly provides connections for the three stepper motors as well as for the several temperature sensors to the main unit. The goniometer main unit contains a Parker Compumotor 6K4 3-axis stepper motor controller, a Lakeshore temperature controller, power supplies, and electrical interfacing. The Compumotor 6K4 controller has both a serial and an Ethernet interface for connection to the personal computer. The Ethernet connection is required for standard operation. Both the Ethernet and the serial connections are required during setup and configuration of a new PC (see Section 4.1).

3.2. Motion Elements

3.2.1. Stepper Motor Controller

The stepper motors and limit switches are controlled via the Parker 6K4 Compumotor controller, which is located inside the goniometer main unit. Communication with the Compumotor controller occurs via an Ethernet LAN. The Ethernet connection is an "Ethernet cross-over cable". This is the only connection required once the system is configured, however a serial connection is needed for the initial setup of the Ethernet connection (see Section 4.1).

3.2.2. Stepper Motors and Rotation Stages

The stepper motors and rotation stages are commercial assemblies purchased from Khozu. Table I, taken from the Khozu product catalog, lists the parameters relevant for the motion control interface:

The stepper motors are <u>not</u> equipped with optical encoders. Absolute positioning therefore has to be implemented by keeping track of all stepper-motor motions relative to a "home position" that is obtained by an initial calibrated homing procedure. The conversion from "number of full stepper motor steps" to "angle" is achieved by multiplying by the angular resolution factors given in Table I (taken directly from the Khozu product catalog).

Note that the **A** stage stepper motor settings differ from those for the **B** and **C** stage stepper motor settings. The **A** stage carries the greatest weight, and high-current high-torque setting used here prevents the **A** stage stepper motor from slipping.

Table I: Khozu Rotation Stage and Stepper Motor Specifications

	A stage	B stage	C stage	Units
Khozu stage model number	RA30A-W	SA16A-RM	SA16A-RT	
Maximum angular rotation	± 155	± 10	± 10	Degrees
Angular resolution	0.002	0.0012	0.00141	degrees per stepper-motor step (full step)
Stepper Motor Setting: Current: Torque:	 5 L/HV on	 3 L/HV off	 3 L/HV off	DIP switches

3.2.3. Limit Switches and DAQ Card

Each rotation stage is equipped with limit switches at both ends of travel. The limit switches for the **B** and **C** stages are mounted rigidly on the stages themselves and cannot be moved. The limit switch for the **A** stage is triggered by aluminum blocks that are mounted on the upper rotating cable tray by means of set screws.

Do not alter the limit switch hardware. The current location of the A-stage blocks is not only set to prevent hardware damage but also set at a location that is used for the homing procedure (see Sections 2.3 and 4.5). Hardware damage can occur and inaccurate homing will result if the A stage limit switch blocks are moved from their current position.

The Compumotor 6K4 controller is hardwired to all six limit switches, which are monitored continuously. Any motion is terminated immediately by the Compumotor 6K4 controller if any of the limit switches is engaged. When this occurs, the only motion that is allowed is a rotation of that axis in the opposite direction.

The limit switches are also monitored by the `GoniometerMotionControl.vi` via a multi-channel DAQ card (installed in the PC) connected to positive end-of-travel limit switches. The limit switches are closed when they are not engaged, and thus the voltage measured across them is zero. When the limit switches are engaged and opened, there is a voltage of ~20 Volts across the respective limit switch connections on the Compumotor 6K4 controller. During the homing procedure, the a LabVIEW sub-VI in conjunction with the DAQ card continuously monitors the voltage across the positive end-of-travel limit switches and issues a HOME=True if all three limit switch voltages exceed a threshold voltage (set to 5 Volts). This is the signal the calibrated movements in the opposite direction towards the "home" position can be initiated.

The limit switches are connected to the DAQ card by means of a custom cable. Figure 2 shows the details of the connections.

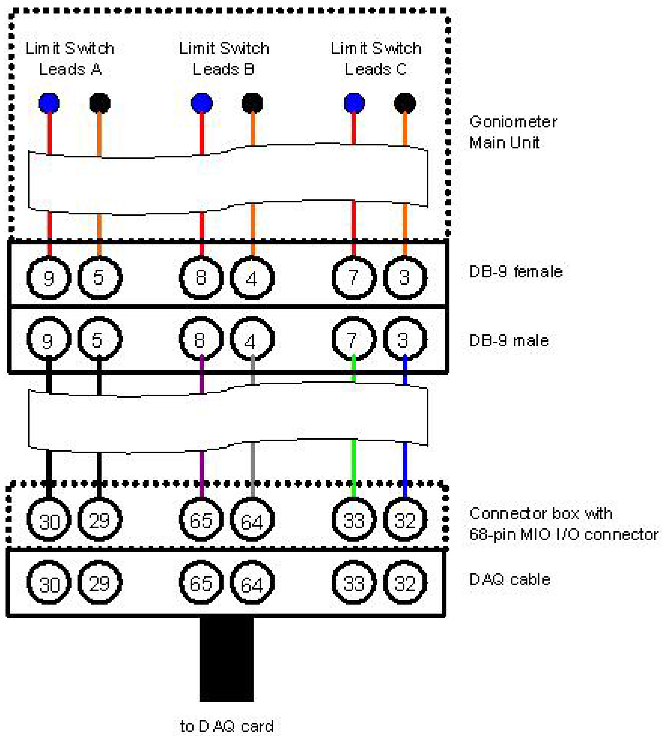


Figure 2: Cable connecting the limit switches inside the goniometer main unit to the DAQ card in the PC. The color coding in the drawing reflects the color coding of the cables.

	The presence of the DAQ card monitoring the limit switches does not alter the continuous, hard-wired limit-switch protection provided by the 6K4 controller. The two are independent. The goniometer limit switches are enforced at all times by the hard-wired limit-switch functionality, independent of any software running on the PC. The use of a separate DAQ card was required due to a problem with the Viewpoint Motion Library that, for some yet unknown reason, prevents reading of the "FastStatus" Information (including limit switch status) via LabVIEW.

4. Software Documentation

4.1. `GoniometerMotionControl.vi` Functionality

The motion control interface `GoniometerMotionControl.vi` for Pharos provides the following basic functionality (see Section 2 for a User Guide):

1. **Home Goniometer:** move each of the three rotation stages to a predefined home position which serves as the reference for all subsequent movements
2. **Move to Pixel:** rotate the goniometer stages such that a diffraction spot that is currently located at a certain position on the Pharos detector bank is moved to a user-specified target position on the Pharos detector bank
3. **Move by Angle:** rotate the goniometer stages by a user-specified relative amount from the current position
4. **Settings:** view system specific parameters

In addition, the motion control software offers the following features:

- All motions are checked against the hardware limits before they are executed. Motions that would drive any of the stages into the respective limit switches are not performed. The only exception is the use of the limit switches as part of the homing procedure (see Section 4.5).
- The software visualizes the current spatial orientation of the three axis of rotation after completion of a motion. Visualization is particularly important since the sample is not visible during an experiment.
- A graph representing the Pharos detector array shows the location of the current and target diffraction sport location after performing a "Move to Pixel" command.

4.2. Stepper Motor Control via Viewpoint Motion Library

The `GoniometerMotionControl.vi` communicates with the Compumotor 6K4 controller via the commercial VI Motion Library by Viewpoint Systems. The Viewpoint Motion Library is a collection of LabVIEW VIs that allows configuring of the 6K4 controller, status monitoring, and execution of motions.

The following is a detailed description of the procedure for setting up the software and communications environment for the Viewpoint Motion Library and the Parker 6K4 Compumotor controller. Note that this is a one-time procedure and has to be worked through only when a new PC is connected to the 6K4 Compumotor controller. All these steps have already been performed for the current Goniometer hardware system in use on Pharos. There are many steps however, and the configuration procedure is shown in detail here for future reference.

The setup procedure comprises three steps:

I. Establishing the Ethernet connection between the PC and the 6K4 Compumotor controller:

1. Connect PC and 6K4 Compumotor controller with an Ethernet cross-over cable.
2. Connect PC COM port to 6K4 Compumotor controller using a RS-232 cable
3. Start Parker Motion Planner software (Version 4.3.0 used)
4. Select Communications – Settings – Port, and then select the COM port used to connect to the 6K4 Compumotor controller. Select OK.
5. Type TNT in the Motion Planner Terminal Window. Status information will appear if the serial connection is established. Otherwise check settings and cabling.
6. Enable Ethernet communications on the controller by typing NTFEN2 in the Motion Planner Terminal Window
7. Set the IP address of the PC's Ethernet card to 192.168.10.31 and network mask 255.255.255.0.
8. Type TNT in the Motion Planner Terminal Window. From the status information, verify that NTFEN2 is enabled. Check that 6K IP address is 192.168.10.30, that the 6K network mask is 255.255.255.0, and that the 6K Ethernet address (hex) is 00-90-55-00-26-FF.
9. Create a 6KARP.BAT file that that will automatically the required ARP –S static mapping procedure. The BAT file contains must be placed into the C:\ root directory and added as "6KARP" to AUTOEXEC.BAT. The 6KARP.BAT file contains one line specifying arp –s "6K IP address" "6K Ethernet address (hex)" "IP address of Ethernet card". In this case:

   ```
   arp -s 192.168.10.30 00-90-55-00-26-FF 192.168.10.31
   ```

10. Exit all programs and restart the PC for the ARP command to be issued upon startup.
11. In Motion Planner, choose Communications – Settings – Port and select Network. In the space following "Network", enter the 6K IP address, i.e. 192.168.10.30, the click OK.
12. In the Motion Planner Terminal Window type TNT. The status should now indicate that Ethernet is connected. At this point, the serial connection between the PC and the 6K4 Compumotor controller is no longer required.

II. Creating the INI file for the Viewpoint 6K Motion Library:

The INI file is used by the `6Kx Comm Launcher.vi` to set up communications, set parameters, and initialize the hardware to a known state. The INI file used here is called "6K_Config_Gonio.INI" and must be placed in the same directory as the main `GoniometerMotionControl.vi`. The INI file looks as follows:

```
[6K_Motion]
6K_Program1="C:\Documents and Settings\robert\My
Documents\Goniometer VI\gonio_02.prg"
6K_Addr1=192.168.10.30,4,10,0
6K_ARP=C:\6KARP.bat
6K_Boxes=1
6K_Setup1=SETUP
Gem6K=1
```

The 6K_Program1 command specifies the path and name of the Motion Planner PRG file that is downloaded to the 6K controller (see step III below). The 6K_Addr1 command specifies the 6K IP address, the number of axes of the controller, the fast status update rate in milliseconds (never use less than 10), and a 0 for a 6K controller. The 6K_ARP command specifies the location of the BAT file with the ARP command. The 6K_Boxes is the number of 6K controllers connected via Ethernet to this PC. 6K_Setup1 specifies the program label that is to be executed in the PRG file to configure, setup or initialize the motion system (see step III below).

III. Creating the PRG file for the 6K4 Compumotor controller:

The Viewpoint VI Motion Library downloads a PRG file to the 6K4 Compumotor controller when the `GoniometerMotionControl.vi` is started up. The PRG file is a collection of low-level 6K4 commands that define the setup of the motion system. The PRG file can also contain low-level functions that enable motion of the motors. The PRG file used here has a "Setup" function that is called by the Viewpoint VI Motion Library to initialize the motion system. This is the primary reason for the existence of the PRG file. The PRG file however also contains functions to move the three motors (A, B, C) in positive or negative directions. These low-level functions can be issued directly from Motion Planner and are useful for basic checks and troubleshooting. During execution of the main `GoniometerMotionControl.vi` application, specific motions do not use these low-level move functions, but rather the Viewpoint VI Motion Library issues the desired motion commands directly.

As specified in step II above, the PRG file is called Gonio_02.PRG and is located in the "`C:\Documents and Settings\robert\My Documents\Goniometer VI`" directory. The PRG file has been adapted from existing Motion Planner code, and a complete listing is given in Appendix A.

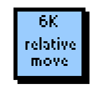
4.3. Structure of `GoniometerMotionControl.vi`

The code consists of a main LabVIEW 7 Express based Virtual Instrument (VI) called `GoniometerMotionControl.vi`. The VI first executes a sequence containing various hardware and software initialization steps. It then enters a State Machine structure represented by a tab control. The "front panel" is the idle case from where specific functions can be called by clicking the respective tabs.

The main VI contains two types of sub-VIs:

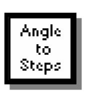
1. LabVIEW-based VIs, including some VIs from the commercially available 6K Motion Library by Viewpoint Systems (with small adaptations)
2. Visual C++-based external functions compiled as dynamic link libraries (DLLs) and called from LabVIEW as Call Library Function Nodes.

The following sections provide detailed descriptions of the more complex sub-VIs. The other sub-VIs should be self-explanatory.



4.4. 6K Relative Move VI

The "`6K Simple Move Basic_V3.vi`" is a sub-VI for executing the motion commands of both the "Home Goniometer" and the "Move by Angle" functions. This VI has been derived from the "`Simple 6K Move Basic.vi`" that is supplied as part of the Viewpoint 6K Motion Library. It (1) issues the commands to the 6K Controller to rotate the stages by a user-specified angle and (2) waits until the commanded motion has been completed **or** until all three positive-anglse limit switches have been engaged. Note that this VI does not check for the angle limits on the stages, i.e. it can be used to drive the stages into their limit switches (a feature that is used by the "Home Goniometer" function). The three axes can be moved simultaneously if desired.



The "`6K Simple Move Basic_V3.vi`" calls the "`AngleToSteps.vi`" to convert the angles in degrees to full steps for the respective stepper motors using the conversion factors given by Khozu and shown in Table I (angular resolution).

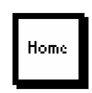
The "`Angle to Steps.vi`" in turn calls the "`CalculateDelay.vi`" which calculates the time it takes for the commanded motion(s) to complete. The three stages may take different times, and the returned delay is the greatest of all three times. If all angles are zero, the delay is zero as well. The delay is based the fixed velocity, acceleration, and deceleration settings of 0.5 (velocity) and 0.1 (acceleration and deceleration) and calculated from the linear calibration parameters shown in Table II. These settings (at bottom of "`6K Simple Move Basic_V3.vi`" front panel) must **not** be changed for two reasons: (1) the vertical **A** axis motor was found to slip at higher velocities and accelerations, most likely due to the substantial weight of the assembly being rotated by it, and the setting used is conservative, and (2) the calibration parameters for calculating the delay (Table II) would have to change for different values.

Table II: Parameters for linear equation to caculate motion times with a fixed velocity of 0.5 and fixed acceleration/deceleration settings of 0.1. The parameters shown here include a 10% safety margin. Note: take the absolute value of the angle before calculating the delay.

Parameter	A stage	B stage	C stage
a_1 (sec/degree)	0.266	0.44	0.44
a_2 (sec)	5.1	6.1	6.1

Note that there is an unresolved issue with the Viewpoint 6K Motion Library installed on the goniometer PC that prevents LabVIEW from reading the 6K4 controller's "Fast Status Update". The "Fast Status Update" would provide essential information on the motion status of the motors and the status of the limit switches. This information is currently not available. We have implemented two workarounds to solve this problem for the Version 1.0 release:

1. A multi-channel DAQ card has been installed in the PC (see Section 3.2.3). It reads the status of the 3 positive-angle limit switches. This functionality is used by the "Home Goniometer" function. Independent of the DAQ card, all limit switches are still visible at the low level to the 6K controller, and any motion that accidentally (or intentionally) drives the motors into the limit switches is immediately terminated by the 6K controller. Only a subsequent motion in the opposite direction can clear a limit-switch fault.
2. With the fixed acceleration, velocity, and deceleration values it is possible to calculate the time it takes to complete a requested angular move without directly monitoring it. Note that each motion has a small overhead in issuing the motion command, accelerating, and decelerating, while the motion itself occurs at uniform velocity. The time required to complete a motion can therefore be described by the linear function $t = a_1\alpha + a_2$. The parameters for the three axes are summarized in Table II.



4.5. Home VI

This VI moves all three stages to a predefined home position. This task is complicated by the absence of absolute encoders and requires the intentional use of the positive-angle limit switches. This is workable but not very elegant solution. Adding absolute encoders to the motors in a future version of the hardware would be desirable.

The home positions are defined as follows:

- **A stage (vertical):** The goniometer top plate has a well-defined position on the Pharos sample well that is provided by means of conical pins in conjunction with a set of bolts. The home position is defined by the goniometer bracket positioned at 90^o to the left with respect to the Pharos incident neutron beam direction.

- **B (horizontal) and C (50° tilted) stages:** The home position is defined by the stages positioned exactly at their respective mid-point, which is characterized by the moving part of the stage being exactly flush with the static part of the respective stage.

The VI simultaneously drives all three stages into the limit switch by issuing a move command with a positive angle that is larger than the total range of motion of the respective stage. The 6K controller immediately aborts the motion when each of the stages triggers its limit switch. The DAQ card senses when all three stages have engaged their respective limit switch (goniometer is now in a known position) and then sends the commands to moce to the "home" position. The "home" position is reached by simultaneously moving all three stages by a previously calibrated negative angle. The angles used during the "Home Goniometer" function are listed in Table III. It is apparent that the limit switches shall not be moved. While this is difficult for the **B** and **C** stages, the **A** stage limit switch is triggered by a block that is fixed to the upper cable tray by means of a set screw. As the label on the goniometer in this location requests, do not move this block.

Table III: Angles for the homing procedure of the three stages

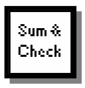
Motion	**A stage**	**B stage**	**C stage**
1. Move into limit switch	$+310^{\circ}$	$+25^{\circ}$	$+25^{\circ}$
2. Move to home position	-89.678°	-10.55°	-10.65°

Note that the stage will be tilted substantially as all three stages drive into their limit switches. The goniometer design does allow for this tilt. Also note that the VI monitors the status of the limit switches and waits until all three limit switches have been engaged. The motion commands #1 are then terminated and the calibrated motion commands #2 (see Table III) are issued.

	The three stages move and tilt substantially as they drive into their limit switches. The goniometer design does allow for this motion. However, be careful not to restrict this motion by any of the sample-related hardware that you mount to the goniometer.

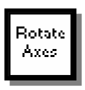
4.6. Move by Angle Function

The "Move by Angle" function on the `GoniometerMotionControl.vi` front panel allows the user to move any or all of the three stages by certain amounts.



The "Move by Angle" function calls the "`SumAndRange.vi`" to calculate the new cumulative angles and to check if the requested motion stays within the angular range for each of the stages. The motion is not performed if any of the stages would exceed its angular range and drive into any of its limit switches.

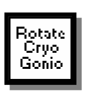
If the motion is in range, the delay time is calculated (see Table II) and the motion is initiated through the "`6K Simple Move Basic_V3.vi`" (see Section 4.4).



Finally, the new orientation of the three axes of rotation in the laboratory frame of reference is calculated by the "`RotateAxes.vi`". This VI issues a call to the "CompoundRotation20.dll" through a LabVIEW Call Library Function. "CompoundRotation20.dll" has been obtained by custom compilation of the Visual C++ code "CompoundRotation.cpp" listed in Appendix B. Please refer to the C++ code for details of the mathematical operations.

4.7. Move to Pixel Function

The "Move to Pixel" function on the `GoniometerMotionControl.vi` front panel allows the user to move the three stages such that a diffraction spot currently located at a certain tube/height pixel on the Pharos detector bank is moved to a new, user-specified tube/height pixel.



At the heart of this function is the "`CallFastRotation.vi`" which solves the inverse rotation problem of having to find three rotation angles given the current and target pixel position of a diffraction spot. The "`CallFastRotation.vi`" issues a call to the "Rotation20.dll" through a LabVIEW Call Library Function. "Rotation20.dll" has been obtained by custom compilation of the Visual C++ code "FastRotation20.cpp" listed in Appendix C. The C++ code provides details of the mathematical operations, but a few notes are given here:

- This task is an "inverse rotation problem" in that the initial and final orientations of the diffraction vector are known, but the three rotation angles producing this compound rotation are unknown. An analytical solution to this problem was not attempted, rather Rotation20.dll numerically scans over all possible A, B, C compound rotations to find the compound rotation that yields the most accurate positioning. See the code listing in the Appendix for the details of the algorithm. Calculation times of a few seconds are typical and acceptable for this application.

- The input to Rotation20.dll is the current orientation of the goniometer as well as the current and target pixel positions of the diffraction spot. On output, Rotation20.dll returns the best α, β, γ rotation angles, the new axis orientations, the new A, B, and

C stage angular positions, and a "Residual" that is a measure of the accuracy of the suggested positioning. "Residual" is the difference between the endpoints of the requested normalized diffraction vector and the best calculated normalized diffraction vector in units of meters. Rotation20.dll returns an error code of 0 if the Residual is less than 0.01 meters, which indicates that the positioning is sufficiently accurate; otherwise an error code of 1 is returned. Inaccurate positioning can be encountered when trying to move a diffraction spot towards the low-angle detectors (tubes number 0-40).

- The function incorporates specifics of the Pharos detector bank geometry, namely the angles for the 376 detector tubes and the distances from the sample to the 376 detector tubes. These values were taken from the existing Pharos IDL data analysis routines.

Appendix A: Parker Motion Planner Program Gonio_02.PRG

```
; WIZID = 00010020
;Product Setup Code
;  Wizard developed for 4 axes with a 6K4 using RS232 COM1

; WIZID = 00010100
;Scaling Setup
;Distance Units - counts,counts,counts,counts
SCLD 1,1,1,1
;Velocity Units - rev/s,rev/s,rev/s,rev/s
SCLV 4000,4000,4000,4000
;Acceleration Units - rev/s/s,rev/s/s,rev/s/s,rev/s/s
SCLA 4000,4000,4000,4000
SCALE1


; WIZID = 00010040
;-------------
;Setup Program
DEL SETUP
DEF SETUP
FOLMAS 0,0,0,0,0,0,0,0
FOLEN00000000

  ; WIZID = 00010060
  ;Enable Mode Code
  DRIVE0000
  ; WIZID = 00010080
  ;Drive Setup
  ;Axis 1, Stepper Control, Other Drive
  ;Axis 2, Stepper Control, Other Drive
  ;Axis 3, Stepper Control, Other Drive
  ;Axis 4, Servo Control, No Drive
  AXSDEF 0001
  DRFLVL 1111
  DRFEN 0000
  KDRIVE XXX0
  DRES 4000,4000,4000,
  PULSE  0.5, 0.5, 2.0,
  DSTALL 000X

  ; WIZID = 00010100
;Scaling Setup
;Because scaling commands are not allowed in a program,
;the scaling commands will be placed at the beginning
;of the program file.  This insures that motion programs
;in subsequent programs will be scaled correctly.

  ; WIZID = 00010120
  ;Feedback Setup
  SFB ,,,1
  ERES ,,,4000
  SMPER ,,,4000
  EFAIL XXX0
  ENCPOL XXX0
  ENCSND XXX0
  ESTALL XXXX
  ESK XXXX
  ENCCNT XXXX
  ; WIZID = 00010140
  ;Hardware Limit Setup
  LH 3,3,3,3
  LHAD 100,100,100,100
  LHADA 100,100,100,100

  ;Software Limit Setup
  LS 0,0,0,0
  LSAD 100,100,100,100
  LSADA 100,100,100,100
  LSNEG 0,0,0,0
  LSPOS 0,0,0,0

  ;Home Limit Setup
  HOMA 10,10,10,10
  HOMAA 10,10,10,10
  HOMV 1,1,1,1
  HOMAD 10,10,10,10
  HOMADA 10,10,10,10
  HOMBAC 0000
  HOMZ 0000
  HOMDF 0000
  HOMVF 0.1,0.1,0.1,0.1
```

```
  HOMEDG 0000

  LIMLVL 000000000000
  ; WIZID = 00010160
  ;Enable Mode Code
  DRIVE1110
END

; WIZID = 00010180
;-------------
;Main Program
DEL MAIN
DEF MAIN

  ; WIZID = 00010200
  GOSUB SETUP
END
STARTP MAIN

; WIZID = 00010220
;-------------
;User Program
DEL ARot
DEF ARot


  ; WIZID = 00010240
  ;Motion Code
  MA 000X
  A 10,10,10,
  AA 0,0,0,
  AD 10,10,10,
  V 0.1,0.1,0.1,
  D -400,400,-400,
  MC 000X

  ; WIZID = 00010260
  ;Go Code
  GO 0010
  ; WIZID = 00010280
  ;Wait Code
  T 3.000


  ; WIZID = 00010300
  BP1
  ;Stop Motion Code

  S
END

; WIZID = 00010320
;-------------
;User Program
DEL AxAPos
DEF AxAPos


  ; WIZID = 00010340
  ;Motion Code
  MA 000X
  A 10,0,0,
  AA 0,0,0,
  AD 10,0,0,
  V 0.1,0,0,
  D 400,0,0,
  MC 000X

  ; WIZID = 00010360
  ;Go Code
  GO 1000
  ; WIZID = 00010380
  ;Wait Code
  T 3.000


  ; WIZID = 00010400
  ;Stop Motion Code

  S
END
```

```
; WIZID = 00010420
;-------------
;User Program
DEL AxANeg
DEF AxANeg


  ; WIZID = 00010440
  ;Motion Code
  MA 000X
  A 10,0,0,
  AA 0,0,0,
  AD 10,0,0,
  V 0.1,0,0,
  D -400,0,0,
  MC 000X

  ; WIZID = 00010460
  ;Go Code
  GO 1000
  ; WIZID = 00010480
  ;Wait Code
  T 3.000


  ; WIZID = 00010500
  ;Stop Motion Code

  S
END

; WIZID = 00010520
;-------------
;User Program
DEL AxBPos
DEF AxBPos


  ; WIZID = 00010540
  ;Motion Code
  MA 000X
  A 0,10,0,
  AA 0,0,0,
  AD 0,10,0,
  V 0,0.1,0,
  D 0,400,0,
  MC 000X

  ; WIZID = 00010560
  ;Go Code
  GO 0100
  ; WIZID = 00010580
  ;Wait Code
  T 3.000


  ; WIZID = 00010600
  ;Stop Motion Code

  S
END

; WIZID = 00010620
;-------------
;User Program
DEL AxBNeg
DEF AxBNeg


  ; WIZID = 00010640
  ;Motion Code
  MA 000X
  A 0,10,0,
  AA 0,0,0,
  AD 0,10,0,
  V 0,0.1,0,
  D 0,-400,0,
  MC 000X

  ; WIZID = 00010660
```

```
  ;Go Code
  GO 0100
  ; WIZID = 00010680
  ;Wait Code
  T 3.000


  ; WIZID = 00010700
  ;Stop Motion Code

  S
END

; WIZID = 00010720
;-------------
;User Program
DEL AxCPos
DEF AxCPos


  ; WIZID = 00010740
  ;Motion Code
  MA 000X
  A 0,0,0.1,
  AA 0,0,0.1,
  AD 0,0,0.1,
  ADA ,,0.1,
  V 0,0,0.25,
  D 0,0,5000,
  MC 000X

  ; WIZID = 00010760
  ;Go Code
  GO 0010
  ; WIZID = 00010780
  ;Wait Code
  T 3.000


  ; WIZID = 00010800
  ;Stop Motion Code

  S
END

; WIZID = 00010820
;-------------
;User Program
DEL AxCNeg
DEF AxCNeg


  ; WIZID = 00010840
  ;Motion Code
  MA 000X
  A 0,0,0.1,
  AA 0,0,0,
  AD 0,0,0.1,
  V 0,0,0.1,
  D 0,0,-10000,
  MC 000X

  ; WIZID = 00010860
  ;Go Code
  GO 0010
  ; WIZID = 00010880
  ;Wait Code
  T 3.000


  ; WIZID = 00010900
  ;Stop Motion Code

  S
END
```

Appendix B: C++ Source Code "CompoundRotation.cpp" used to compile the Dynamic Link Library "CompoundRotation20.dll"

```
// Call Library source file (DLL)

//*****************************************************************************
//
// Revision History:
// 10/5/2004          Version 1.0           Created CompoundRotation from existing
//                                          FastRotation20.cpp
//
//*****************************************************************************
//
//                    PROGRAM FOR CALCULATING A COMPOUND ROTATION
//                 FOR THE CALTECH/CHAVE 3-AXIS CRYOGENIC GONIOMETER
//                 -------------------------------------------------
//
//                      Markus P. Hehlen  (hehlen@lanl.gov)
//                        Los Alamos National Laboratory,
//                   LANSCE-12, MS H805, Los Alamos, NM 87545
//
//
// The code takes an initial configuration of the three goniometer axes A, B,
// and C, then performs a compound rotation A(alpha), B(beta), C(gamma), and
// as a result returns the new configuration of the three goniometer axes.
//
// The conventions for the coordinate system are as follows:
//   - the X axis points right (looking in te direction of the incoming neutron
//     beam),
//   - the Y-axis points up (in the lab),
//   - and the Z-axis points inward (in the direction of the incoming neutron
//     beam).
//
// Therefore,
//   - the XZ plane is "on the floor", i.e. this is the horizontal scattering
//     plane
//   - the YZ plane is a "side-wall",
//   - and the XY plane is a "front-wall."
//
// It is also assumed that the incident neutron beam propagates along the
// z-axis, i.e. (0,0,1). The origin of this coordinate system is at the sample
// location, i.e. at the point where the incident neutron beam intersects the
// vertical axis of the goniometer.


#include "C:\Program Files\National Instruments\LabVIEW 7.0\cintools\extcode.h"


// Declare the external function
// Note: The extern "C" prevents the C++ compiler from decorating the function
// names in the final object code. Also, the _declspec(dllexport) keyword is
// needed to explicitly export the function from the DLL to make it
// available to LabVIEW.
extern "C" {
  _declspec(dllexport) long CompoundRotation(double *AxisA, double *AxisB,
  double *AxisC, double alpha, double beta, double gamma, double *NewAx,
  double *NewAy, double *NewAz, double *NewBx, double *NewBy, double *NewBz,
  double *NewCx, double *NewCy, double *NewCz);
} // extern "C"


#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <stdlib.h>

using namespace std;


//*****************************************************************************
//****************************** GLOBAL DEFINITIONS ****************************
//*****************************************************************************

const double PI180=M_PI/180;
const double invPI180=1/PI180;
const double PI=M_PI;

// define a 3D vector structure
struct vector {
  double x,y,z;
};
```

```
// define a line
struct ArbAxis {
  vector begin,end;
};



//*********************************************************************************
//*********************** INTERNAL (not exported) FUNCTIONS ********************
//*********************************************************************************

void rotate (vector& P, ArbAxis L, double theta) {
// Rotate point P about arbitrary line L (defined by end points A and B) by
// angle theta (in rad). The code contained in this function was takedn
// directly from http://www-scf.usc.edu/~akotaobi/gptut13_1.html. The function
// returns the rotated coordinates of P.

  vector A,B,A2,B2;
  double tempx, tempy, tempz;
  double len1, len2;
  double costheta1, sintheta1,costheta2, sintheta2;
  A=L.begin;
  B=L.end;

  // 1. Create a copy of the line for manipulation.
  A2=A;
  B2=B;

  // 2. Translate L so one endpoint lies at the origin, i.e. translate P by -A
  P.x -= A.x;
  P.y -= A.y;
  P.z -= A.z;
  // Translate copy of line similarly
  A2.x -= A.x;
  A2.y -= A.y;
  A2.z -= A.z;
  B2.x -= A.x;
  B2.y -= A.y;
  B2.z -= A.z;

  // 3. Rotate around the Y-axis until L lies in the YZ plane.
  len1 = sqrt(B2.x * B2.x + B2.z * B2.z);
  if (len1 > 0) {
    double invlen1=1/len1;
        costheta1 = B2.z * invlen1;
    sintheta1 = B2.x * invlen1;
    // Apply rotation to P
    tempx = P.x * costheta1 - P.z * sintheta1;
    tempz = P.x * sintheta1 + P.z * costheta1;
    P.x = tempx;
    // P.y remains constant
    P.z = tempz;
    // Apply rotation to line copy
    tempx = B2.x * costheta1 - B2.z * sintheta1;
    tempz = B2.x * sintheta1 + B2.z * costheta1;
    B2.x = tempx;
    // B2.y remains constant
    B2.z = tempz;
  }

  // 4. Rotate around the X axis until L lies on the Z axis.
  len2 = sqrt(B2.y * B2.y + B2.z * B2.z);
  if (len2 > 0) {  // Should always be true
    double invlen2=1/len2;
        costheta2 = B2.z * invlen2;
    sintheta2 = B2.y * invlen2;
    // Apply rotation to P
    tempy = P.y * costheta2 - P.z * sintheta2;
    tempz = P.y * sintheta2 + P.z * costheta2;
    // P.x remains constant
    P.y = tempy;
    P.z = tempz;
  }

  // 5. Perform the rotation around the z-axis.
  double ctheta=cos(theta);
  double stheta=sin(theta);
  tempx = P.x * ctheta - P.y * stheta;
  tempy = P.x * stheta + P.y * ctheta;
  P.x = tempx;
```

```
    P.y = tempy;
    // P.z remains constant

    // 6. Undo Step 4.
    if (len2 > 0) {  // Should always be true
      // Apply rotation to P only
      tempy = P.y * costheta2 + P.z * sintheta2;
      tempz = -P.y * sintheta2 + P.z * costheta2;
      P.y = tempy;
      P.z = tempz;
    }

    // 7. Undo Step 3.
    if (len1 > 0) { // If we never performed step 3, don't perform this step
      // Apply rotation to P only
      double tempx = P.x * costheta1 + P.z * sintheta1;
      double tempz = -P.x * sintheta1 + P.z * costheta1;
      P.x = tempx;
      P.z = tempz;
    }

    // 8. Undo Step 2.
    P.x += A.x;
    P.y += A.y;
    P.z += A.z;
  } // rotate


  //*******************************************************************************
  //************* DEFINITION OF FUNCTION TO BE EXPORTED FROM THIS DLL ************
  //*******************************************************************************



  _declspec(dllexport) long CompoundRotation(double *AxisA, double *AxisB,
          double *AxisC, double alpha, double beta, double gamma, double *NewAx,
          double *NewAy, double *NewAz, double *NewBx, double *NewBy, double *NewBz,
          double *NewCx, double *NewCy, double *NewCz)
          {

    // conversion to rad
    alpha=alpha*PI180;
    beta=beta*PI180;
    gamma=gamma*PI180;

    // define goniometer axes from the coordinates passed
    ArbAxis A,B,C;
    A.begin.x=0;          A.begin.y=0;          A.begin.z=0;
    A.end.x=AxisA[0];     A.end.y=AxisA[1];     A.end.z=AxisA[2];
    B.begin.x=0;          B.begin.y=0;          B.begin.z=0;
    B.end.x=AxisB[0];     B.end.y=AxisB[1];     B.end.z=AxisB[2];
    C.begin.x=0;          C.begin.y=0;          C.begin.z=0;
    C.end.x=AxisC[0]; C.end.y=AxisC[1];C.end.z=AxisC[2];

    // Calculate and return new orientations of three axes
    ArbAxis A_new,B_new,C_new;
    // 1. The A(alpha) axis cannot change since it is fixed in space
    A_new=A;
    // 2. The B(beta) axis was rotated about A by alpha
    B_new=B;
    rotate (B_new.begin,A_new,alpha);
    rotate (B_new.end,A_new,alpha);
    // 3. The C(gamma) axis was rotated about A by alpha and about B by beta
    C_new=C;
    rotate (C_new.begin,A_new,alpha);
    rotate (C_new.end,A_new,alpha);
    rotate (C_new.begin,B_new,beta);
    rotate (C_new.end,B_new,beta);
    // 4. return values
    *NewAx=A_new.end.x;   *NewAy=A_new.end.y;   *NewAz=A_new.end.z;
    *NewBx=B_new.end.x;   *NewBy=B_new.end.y;   *NewBz=B_new.end.z;
    *NewCx=C_new.end.x;   *NewCy=C_new.end.y;   *NewCz=C_new.end.z;

    return(0);

  }  // CompoundRotation
```

Appendix C: C++ Source Code "FastRotation20.cpp" used to compile the Dynamic Link Library "Rotation20.dll"

```
// Call Library source file (DLL)

//*******************************************************************************
//
// Revision History:
// 9/20/2004          Version 1.0          FastRotation.cpp; Initial release.
// 9/23/2004          Version 1.1          Ported to Microsoft Viasual C++ Development
//                                         Environment 2003 (7.1.3088).
// 9/29/2004          Version 1.2          Correction of rotation algorithm through
//                                         implementation of surface normal rotation.
//                                         FastRotation12.cpp
// 10/1/2004          Version 2.0          Converted FastRotation12.cpp to a DLL for
//                                         enableing its execution from LabView through
//                                         a Call Library Function Node.
//
//*******************************************************************************
//
//                    PROGRAM FOR CALCULATING ROTATIONS FOR THE
//                    CALTECH/CHAVE 3-AXIS CRYOGENIC GONIOMETER
//                    -----------------------------------------
//
//                       Markus P. Hehlen  (hehlen@lanl.gov)
//                         Los Alamos National Laboratory,
//                    LANSCE-12, MS H805, Los Alamos, NM 87545
//
//
// The code defines a starting configuration for the three goniometer axes and a
// current pixel location of a diffraction spot on the Pharos detector array.
// The user then specifies a target pixel to which the diffraction spot has to
// be moved.  The code calculates the surface normals on both the current
// diffraction plane and the target diffraction plane. The current and target
// diffraction spot determine the rotation direction for the A and B axes (but
// not the dependent C axis), elminating half of the possible angles. The code
// then iterates over a grid of angles in the respective angle ranges of the
// three axes and searches for the rotation that most closely moves the current
// surface normal into the target surface normal. The rotation sequence is fixed
// as A(alpha)->B(beta)->C(gamma) for which a solution can always be found.
//
// The conventions for the coordinate system are as follows:
//   - the X axis points right (looking in te direction of the incoming neutron
//     beam),
//   - the Y-axis points up (in the lab),
//   - and the Z-axis points inward (in the direction of the incoming neutron
//     beam).
//
// Therefore,
//   - the XZ plane is "on the floor", i.e. this is the horizontal scattering
//     plane
//   - the YZ plane is a "side-wall",
//   - and the XY plane is a "front-wall."
//
// It is also assumed that the incident neutron beam propagates along the
// z-axis, i.e. (0,0,1). The origin of this coordinate system is at the sample
// location, i.e. at the point where the incident neutron beam intersects the \
// vertical axis of the goniometer.


#include "C:\Program Files\National Instruments\LabVIEW 7.0\cintools\extcode.h"

// Declare the external function
// Note: The extern "C" prevents the C++ compiler from decorating the function
// names in the final object code. Also, the _declspec(dllexport) keyword is
// needed to explicitly export the function from the DLL to make it
// available to LabVIEW.
extern "C" {
  _declspec(dllexport) long FastRotation(double *AxisA, double *AxisB,
         double *AxisC, double BracketAngle, double Arange, double Brange,
         double Crange, double SumA, double SumB, double SumC, long CurrentTube,
         long CurrentHeight, long TargetTube, long TargetHeight, double *BestA,
         double *BestB, double *BestC, double *Residual, double *NewAx,
         double *NewAy, double *NewAz, double *NewBx, double *NewBy,
         double *NewBz, double *NewCx, double *NewCy, double *NewCz, long *Error);
}  // extern "C"

#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <fstream>
#include <stdlib.h>
```

```
using namespace std;

//******************************************************************************
//***************************** GLOBAL DEFINITIONS *****************************
//******************************************************************************

const double PI180=M_PI/180;
const double invPI180=1/PI180;
const double PI=M_PI;

// define a 3D vector structure
struct vector {
  double x,y,z;
};

// define a line
struct ArbAxis {
  vector begin,end;
};



//******************************************************************************
//*********************** INTERNAL (not exported) FUNCTIONS ********************
//******************************************************************************


void NormalizeVector (vector& P) {
// Normalizes vector P
  double INVlen=1/sqrt(P.x*P.x+P.y*P.y+P.z*P.z);
  P.x=P.x*INVlen;
  P.y=P.y*INVlen;
  P.z=P.z*INVlen;
}


//******************************************************************************


double PharosAngle (int tube) {
// calculates the angle in degrees of Pharos tube number "tube" in the
// horizontal (xz) plane. Formula taken from Pharos IDL routines.
  double detang;
  if (tube<=23) {
    detang=-10.9 + 0.4*tube;
  }
  else {
    if (tube<=39) {
      detang=-7.83 + 0.4*tube;
        }
        else {
          if (tube<=151) {
            detang=0.4011*tube - 6.1665;
          }
          else {
               if (tube<=263) {
          detang=0.4029*tube - 5.6868;
               }
               else {
          detang=0.412*tube - 7.4276;
               }
          }
        }
  }
  return detang;
} // PharosAngle


//******************************************************************************


double PharosDistance (int tube) {
// calculates the distance in meters from the sample to the center of Pharos
// tube number "tube". Again, the distance is in the horizontal (x,z) plane. The
// formula ios from the Pharos IDL routines without the "fudge factor" (which is
// for the old RT stick only.
  double dist;
  if (tube<=39) {
    dist=3.972/cos(PharosAngle(tube)*PI180);
  }
  else {
```

```
    dist=4.013;
  }
  return dist;
} // PharosDistance


//********************************************************************************


void PharosPixel (vector p, int PixelDensity, int& tube, int& height) {
// takes vector p (originating at origin of coordinate system, i.e. the sample)
// and calculates which pixel in the Pharos detector bank it is pointing to. The
// function utilizes the Pharos detector bank angular and distance calibration
// from the Pharos IDL routines. The tube is assumed to be 1.0 meters long and
// has PixelDensity pixles per meter.

  // precalculate some numbers for increased speed
  double px2=p.x*p.x;
  double py2=p.y*p.y;
  double pz2=p.z*p.z;
  double sum_xz=px2+pz2;

  // 1. Obtain pointing angle of p in the xz (horizontal) plane
  double xz_angle;
  if ( p.z==0 ) { xz_angle=0.5*PI; }
  else {
    xz_angle=atan(p.x/p.z);
    if (xz_angle<0) { xz_angle=-xz_angle; }
  }
  if ( (p.z>0) && (p.x<0) ) { xz_angle=-xz_angle; }
  else {
    if (p.z<0) { xz_angle=PI-xz_angle; }
  }
  xz_angle=xz_angle*invPI180;   // conversion to degrees

  // 2. Find the detector tube for which xz_angle most closely matches
  // Note: the loop is executed only as long as diff improves, and the loops
  // breaks as soon as diff gets worse. This can be done since there is only one
  // minimum for diff as a function of tube#, which is the location of the tube
  // we are pointing at.
  double diff,min,old_diff;
  min=1.0E10;
  old_diff=1.0E10;
  for (int i=0; i<=375; i++) {
    diff=pow((PharosAngle(i)-xz_angle),2);
        if (diff>old_diff) { break; }  // getting worse, quit this.
        old_diff=diff;
        if (diff<min) {
      min=diff;
          tube=i;
        }
  }
  if (xz_angle<PharosAngle(0)) {
          // scattered too far to the left and missed the detector bank
          tube=-1000;
  }
  if (xz_angle>PharosAngle(375)) {
          // scattered too far to the right and missed the detector bank
          tube=+1000;
  }

  // 3. Find height pixel. Do this only if we hit any of the tubes based on the
  // horizontal scattering angle
  if ((tube<1000) && (tube>-1000)) {
        //out of (xz)-plane angle
        double phixz=acos( (sum_xz)/(sqrt(px2+py2+pz2)*sqrt(sum_xz)) );
        if ((p.y<1E-10) && (p.y>-1E-10)) {
                // catch acos floating point overflow in case phixz is zero or near-zero
                phixz=0;
        }
        double x=sin(phixz)*PharosDistance(tube);
        if (p.y<0) {
                // scattered downward, i.e. the vertical distance x on the tube is
                // negative
                x=-x;
        }
        // calculate height pixel from distance. Note: pixel numbering starts at the
        // bottom of the tube
        height=(x*PixelDensity)+0.5*PixelDensity;
        if (height<0) {
                // scattered downward too much and missed detector bank
```

```
            height=-1000;
        }
        if (height>40) {
            // scattered upward too much and missed detector bank
            height=1000;
        }
    }
    else {
        // we didn't hit the detector bank in the horizontal plane in the first
        // place. So assign and error number to height as well
        height=-1001;
    }
} // PharosPixel

//*******************************************************************************

void SurfaceNormal (vector In, vector Out, vector& N) {
// Given an incoming (In) and outgoing (Out) vector, the function calculates the
// vector N that is normal on the respective scattering plane. Both In and Out
// are first normalized. The In vector points from a point P1 to the origin
// (0,0,0) and the Out vector points from the origin (0,0,0) to a point P2.
// Point P is halfway between P1 and P2, and the vector going from the origin
// through P is coaxial with the surface normal. The final surface normal vector
// N is obtained by normalizing the vector O->P.
  // Normalize In and Out
  NormalizeVector(In);
  NormalizeVector(Out);
  // Find P and normalize
  N.x=0.5*(-In.x+Out.x);
  N.y=0.5*(-In.y+Out.y);
  N.z=0.5*(-In.z+Out.z);
  NormalizeVector(N);
} // SurfaceNormal

//*******************************************************************************

void rotate (vector& P, ArbAxis L, double theta) {
// Rotate point P about arbitrary line L (defined by end points A and B) by
// angle theta (in rad). The code contained in this function was takedn
// directly from http://www-scf.usc.edu/~akotaobi/gptut13_1.html. The function
// returns the rotated coordinates of P.

  vector A,B,A2,B2;
  double tempx, tempy, tempz;
  double len1, len2;
  double costheta1, sintheta1,costheta2, sintheta2;
  A=L.begin;
  B=L.end;

  // 1. Create a copy of the line for manipulation.
  A2=A;
  B2=B;

  // 2. Translate L so one endpoint lies at the origin, i.e. translate P by -A
  P.x -= A.x;
  P.y -= A.y;
  P.z -= A.z;
  // Translate copy of line similarly
  A2.x -= A.x;
  A2.y -= A.y;
  A2.z -= A.z;
  B2.x -= A.x;
  B2.y -= A.y;
  B2.z -= A.z;

  // 3. Rotate around the Y-axis until L lies in the YZ plane.
  len1 = sqrt(B2.x * B2.x + B2.z * B2.z);
  if (len1 > 0) {
    double invlen1=1/len1;
        costheta1 = B2.z * invlen1;
    sintheta1 = B2.x * invlen1;
    // Apply rotation to P
    tempx = P.x * costheta1 - P.z * sintheta1;
    tempz = P.x * sintheta1 + P.z * costheta1;
    P.x = tempx;
    // P.y remains constant
    P.z = tempz;
```

```
    // Apply rotation to line copy
    tempx = B2.x * costheta1 - B2.z * sintheta1;
    tempz = B2.x * sintheta1 + B2.z * costheta1;
    B2.x = tempx;
    // B2.y remains constant
    B2.z = tempz;
  }

  // 4. Rotate around the X axis until L lies on the Z axis.
  len2 = sqrt(B2.y * B2.y + B2.z * B2.z);
  if (len2 > 0) {  // Should always be true
    double invlen2=1/len2;
        costheta2 = B2.z * invlen2;
    sintheta2 = B2.y * invlen2;
    // Apply rotation to P
    tempy = P.y * costheta2 - P.z * sintheta2;
    tempz = P.y * sintheta2 + P.z * costheta2;
    // P.x remains constant
    P.y = tempy;
    P.z = tempz;
  }

  // 5. Perform the rotation around the z-axis.
  double ctheta=cos(theta);
  double stheta=sin(theta);
  tempx = P.x * ctheta - P.y * stheta;
  tempy = P.x * stheta + P.y * ctheta;
  P.x = tempx;
  P.y = tempy;
  // P.z remains constant

  // 6. Undo Step 4.
  if (len2 > 0) {  // Should always be true
    // Apply rotation to P only
    tempy = P.y * costheta2 + P.z * sintheta2;
    tempz = -P.y * sintheta2 + P.z * costheta2;
    P.y = tempy;
    P.z = tempz;
  }

  // 7. Undo Step 3.
  if (len1 > 0) { // If we never performed step 3, don't perform this step
    // Apply rotation to P only
    double tempx = P.x * costheta1 + P.z * sintheta1;
    double tempz = -P.x * sintheta1 + P.z * costheta1;
    P.x = tempx;
    P.z = tempz;
  }

  // 8. Undo Step 2.
  P.x += A.x;
  P.y += A.y;
  P.z += A.z;
} // rotate


//*******************************************************************************


void CompoundRotation (vector& P, ArbAxis A, double alpha, ArbAxis B,
                                        double beta, ArbAxis C, double gamma) {
// Compound rotation of point P about axes A, B, C by alpha, beta gamma (in
// rad), respectively, in the predefined sequence A(alpha)->B(beta)->C(gamma).
// There is an important subtlety about the way the goniometer is built. It is
// instructive to remember that the motors are stacked and the implications of
// this. Therefore:
//   if I rotate about A I will rotate both B and C with it
//   if I rotate about B I will not change A but I will rotate C with it
//   if I rotate about C I will not change A and will not change B
// Also note that even if I rotate A, rotating about the B axis will still
// perform an up (beta>0) or down (beta<0) motion of the diffracted spot. This
// property is useful for predetermining the sign of beta for a particular
// motion of a diffraction spot. In contrast, the C-axis can be skewed by
// rotations about B and its effect on moving a diffraction spot are thus
// dependent on the orientation of B.

  // 1. Rotate point P and begin & end points of ArbAxis B and C about ArbAxis
  // A by alpha
  rotate(P,A,alpha);
  rotate(B.begin,A,alpha);
  rotate(B.end,A,alpha);
```

```
  rotate(C.begin,A,alpha);
  rotate(C.end,A,alpha);
  // 2. Rotate point P and begin & end points of ArbAxis C about ArbAxis B
  // by beta
  rotate(P,B,beta);
  rotate(C.begin,B,beta);
  rotate(C.end,B,beta);
  // 3. Rotate point P about ArbAxis C by gamma
  rotate(P,C,gamma);
} // CompoundRotation


//******************************************************************************


void VectorFromPixel (int tube, int height, int PixelDensity, vector& P) {
// calculates the vector P that points from the sample to a pixel at
// (tube,height)
  // 1. Calculate Pharos detector angles; taken from IDL routine DETECTORANGLE
  double phi=PharosAngle(tube)*PI180;    // in rad
  // 2. Generate a vector that point towards that tube (in horizontal plane);
  vector Pc;
  Pc.x=0;  Pc.y=0;  Pc.z=1;
  ArbAxis L;
  L.begin.x=0; L.begin.y=0;   L.begin.z=0;
  L.end.x=0;   L.end.y=1;             L.end.z=0;
  rotate(Pc,L,phi);
  // calculate distance along tube. Note the -0.5 in the 1st parenthesis. This
  // takes care of the fact that the center of the tube (Pc.y=0) lies exactly
  // between height pixel #19 and #20 (since pixel numbering starts at the
  // bottom with #0). So the center is at pixel number "19.5", which is 0.5 less
  // than half the number of pixels on the tube, i.e. 0.5*PixelDensity-0.5 .
  Pc.y= (height-(0.5*PixelDensity-0.5))/PixelDensity;
  // scale Pc.x and Pc.z such that they amount to the distance to the tube in
  // the xz-plane (horizontal)
  double a=sqrt(pow(PharosDistance(tube),2)/(Pc.x*Pc.x+Pc.z*Pc.z));
  Pc.x=Pc.x*a;
  Pc.z=Pc.z*a;
  // 3. Normalize vector P
  NormalizeVector(Pc);
  P=Pc;
} // VectorFromPixel


//******************************************************************************
//************* DEFINITION OF FUNCTION TO BE EXPORTED FROM THIS DLL ************
//******************************************************************************


_declspec(dllexport) long FastRotation(double *AxisA, double *AxisB,
    double *AxisC, double BracketAngle, double Arange, double Brange,
        double Crange, double SumA, double SumB, double SumC, long CurrentTube,
        long CurrentHeight, long TargetTube, long TargetHeight, double *BestA,
        double *BestB, double *BestC, double *Residual, double *NewAx,
        double *NewAy, double *NewAz, double *NewBx, double *NewBy,
        double *NewBz, double *NewCx, double *NewCy, double *NewCz, long *Error) {

               // Definition of the function parameters:
// double AxisA[]            Initial A axis configuration, 3-element array (x,y,z)
// double AxisB[]            Initial B axis configuration, 3-element array (x,y,z)
// double AxisC[]            Initial C axis configuration, 3-element array (x,y,z)
// double BracketAngle Angle of the C-stage bracket in degrees (50.0)
// double Arange             Maximum +/- angular range of A axis in degrees (155.0)
// double Brange             Maximum +/- angular range of B axis in degrees (10.0)
// double Crange             Maximum +/- angular range of C axis in degrees (10.0)
// double SumA               Initial cumulative angular position of A axis in degrees
// double SumB               Initial cumulative angular position of B axis in degrees
// double SumC               Initial cumulative angular position of C axis in degrees
// long CurrentTube          Tube# of current diffraction sport
// long CurrentHeight  Height pixel# of current diffraction spot
// long TargetTube           Tube# of target diffraction spot
// long TargetHeight   Height pixel# of target diffraction spot
// double *BestA             Best rotation angle alpha in degrees
// double *BestB             Best rotation angle beta in degrees
// double *BestC             Best rotation angle gamma in degrees
// double *Residual          Distance between endpoints of rotated current surface
//                                  normal and target surface normal
// double NewA[]             New A axis configuration, 3-element array (x,y,z)
// double NewB[]             New B axis configuration, 3-element array (x,y,z)
// double NewC[]             New C axis configuration, 3-element array (x,y,z)
// long *Error               Error code (0=no error)
```

```
//
//*******************************************************************************

  // define goniometer axes from the coordinates passed
  ArbAxis A,B,C;
  A.begin.x=0;         A.begin.y=0;          A.begin.z=0;
  A.end.x=AxisA[0];    A.end.y=AxisA[1];     A.end.z=AxisA[2];
  B.begin.x=0;         B.begin.y=0;          B.begin.z=0;
  B.end.x=AxisB[0];    B.end.y=AxisB[1];     B.end.z=AxisB[2];
  C.begin.x=0;         C.begin.y=0;          C.begin.z=0;
  C.end.x=AxisC[0]; C.end.y=AxisC[1]; C.end.z=AxisC[2];

  // define maximum angular range ( +/- the angle)  (in degrees)
  // The full range of the axes is given by the hardware and is alpha_range=155
  // (Khozu RA20), beta_range=gamma_range=10 (Khozu SA16)
  double alpha_range, beta_range, gamma_range;
  alpha_range=Arange;
  beta_range=Brange;
  gamma_range=Crange;
  // convert to rad
  alpha_range=alpha_range*PI180;
  beta_range=beta_range*PI180;
  gamma_range=gamma_range*PI180;

  // convert the passed cumulative angles of the stages to rad
  double alpha_old, beta_old, gamma_old;
  alpha_old=SumA*PI180;
  beta_old=SumB*PI180;
  gamma_old=SumC*PI180;

  // Define normalized incoming beam direction
  vector In;
  In.x=0;
  In.y=0;
  In.z=1;

  // Current diffraction geometry:
  // calculate normalized vector Pc pointing from the sample center to the
  // current (tube,height) pixel. Thus, Pc is the direction of the scattered
  // current beam of interest.
  vector Pc;
  VectorFromPixel(CurrentTube,CurrentHeight,40,Pc);
  // We now have the incoming beam direction and the current scattered beam
  // direction. From these, calculate the surface normal Nc of the current
  // scattering plane.
  vector Nc;
  SurfaceNormal(In,Pc,Nc);

  // Target diffraction geometry:
  // calculate normalized vector Pt pointing from the sample center to the
  // target (tube,height) pixel. Thus, Pt is the direction of the scattered
  // target beam of interest.
  vector Pt;
  VectorFromPixel(TargetTube,TargetHeight,40,Pt);
  // We now have the incoming beam direction and the target scattered beam
  // direction. From these, calculate the surface normal Nt of the target
  // scattering plane (the one that produces the target diffraction spot.
  vector Nt;
  SurfaceNormal(In,Pt,Nt);

  // Now remember: we are working with the surface normals which
  // automatically takes care of the properties of a reflection/diffraction
  // (for example that the spot moves 2theta when an axis is rotated by
  // theta). The task is to find the three goniometer rotation angles such
  // that Nc is rotated into Nt.

  // Based on the current and target pixel, we can determine the direction of
  // rotation of both the A and B axis. This knowledge enables us to eliminate
  // unnecessary iterations over rotations that go in the wrong direction,
  // thereby increasing computational speed.
  // define rotation direction for A(alpha) axis: Moving to lower tube numbers
  // is a positive alpha
  int i_start,i_stop;
  if (CurrentTube>=TargetTube) {
       // the move is from a higher tube# to a lower tube#, thus only scan over
       //positive alpha
       i_start=0;
       i_stop=50;
  }
  else {
       // the move is from a lower tube# to a higher tube#, thus only scan over
```

```
      // negative alpha
      i_start=-50;
      i_stop=0;
}
// define rotation direction for B(beta) axis: Moving to a larger height
// (up) is a positive beta
int j_start,j_stop;
if (CurrentHeight>=TargetHeight) {
      // the move is from up towards down in height, thus only scan over
      // negative beta
      j_start=-50;
      j_stop=0;
}
else {
      // the move is from down towards up in height, thus only scan over
      // positive beta
      j_start=0;
      j_stop=50;
}
// Note: we are not restricting the motion of the C axis (see notes in
// definition of function CompoundRotation(). Both positive and negative
// gammas shall be possible in all circumstances.

// definitions for iterations
bool OutOfRange;
double alpha_min, beta_min, gamma_min, dist_min, dist2;
dist_min=1E10;
double alpha,beta,gamma;
vector P;

// start iterations
for (int i=i_start; i<=i_stop; i++) {       // over alpha
  for (int j=j_start; j<=j_stop; j++) {     // over beta
    for (int k=-50; k<=50; k++) {           // over gamma

             // define angles of rotation based on the predefined direction and
             // the full range of motion
         alpha = alpha_range * (i*0.02);
         beta  = beta_range  * (j*0.02);
         gamma = gamma_range * (k*0.02);

             // eliminate any rotations that happen to be beyond the possible range
             // of motion for the stages
             OutOfRange=false;
             if ( ((alpha_old+alpha)>alpha_range) ||
                  ((alpha_old+alpha)<-alpha_range) ) { OutOfRange=true; }
             if ( ((beta_old+beta)>beta_range)    ||
                  ((beta_old+beta)<-beta_range)    ) { OutOfRange=true; }
             if ( ((gamma_old+gamma)>gamma_range) ||
                  ((gamma_old+gamma)<-gamma_range) ) { OutOfRange=true; }

             if (OutOfRange==false) {
               // only perform compound rotation if none of the angles gets any of
               // the stages out of range. Do compound rotation 1->2->3 with above
               // angles. First, define the point to be rotated; it is the end
               // point of the current surface normal vector
            P=NC;
               // now rotate
            CompoundRotation(P,A,alpha,B,beta,C,gamma);
               // calculate (squared) distance from rotated surface normal vector
               // to the end point of the target surface normal vector
        dist2=pow((P.x-Nt.x),2) + pow((P.y-Nt.y),2) + pow((P.z-Nt.z),2);
               if (dist2<dist_min) {
          // found new best distance. Record the new minimum distance and
                     // the angles
          dist_min=dist2;
                 alpha_min=alpha;
                 beta_min=beta;
                 gamma_min=gamma;
               }
             }
       }
  }
}

// Return best rotation
*BestA=alpha_min*invPI180;
*BestB=beta_min*invPI180;
*BestC=gamma_min*invPI180;
*Residual=sqrt(dist_min);
```

```
    // Calculate and return new orientations of three axes
    ArbAxis A_new,B_new,C_new;
    // 1. The A(alpha) axis cannot change since it is fixed in space
    A_new=A;
    // 2. The B(beta) axis was rotated about A by alpha
    B_new=B;
    rotate (B_new.begin,A_new,alpha_min);
    rotate (B_new.end,A_new,alpha_min);
    // 3. The C(gamma) axis was rotated about A by alpha and about B by beta
    C_new=C;
    rotate (C_new.begin,A_new,alpha_min);
    rotate (C_new.end,A_new,alpha_min);
    rotate (C_new.begin,B_new,beta_min);
    rotate (C_new.end,B_new,beta_min);
    // 4. return values
    *NewAx=A_new.end.x;   *NewAy=A_new.end.y;   *NewAz=A_new.end.z;
    *NewBx=B_new.end.x;   *NewBy=B_new.end.y;   *NewBz=B_new.end.z;
    *NewCx=C_new.end.x;   *NewCy=C_new.end.y;   *NewCz=C_new.end.z;

    // Assign Error code
    if (sqrt(dist_min)<0.01) {
         // accurate positioning expected
      *Error=0;
    }
    else {
         // Residual large: expect inaccurate positioning
         *Error=1;
    }

    return(0);
  } // FastRotation
```

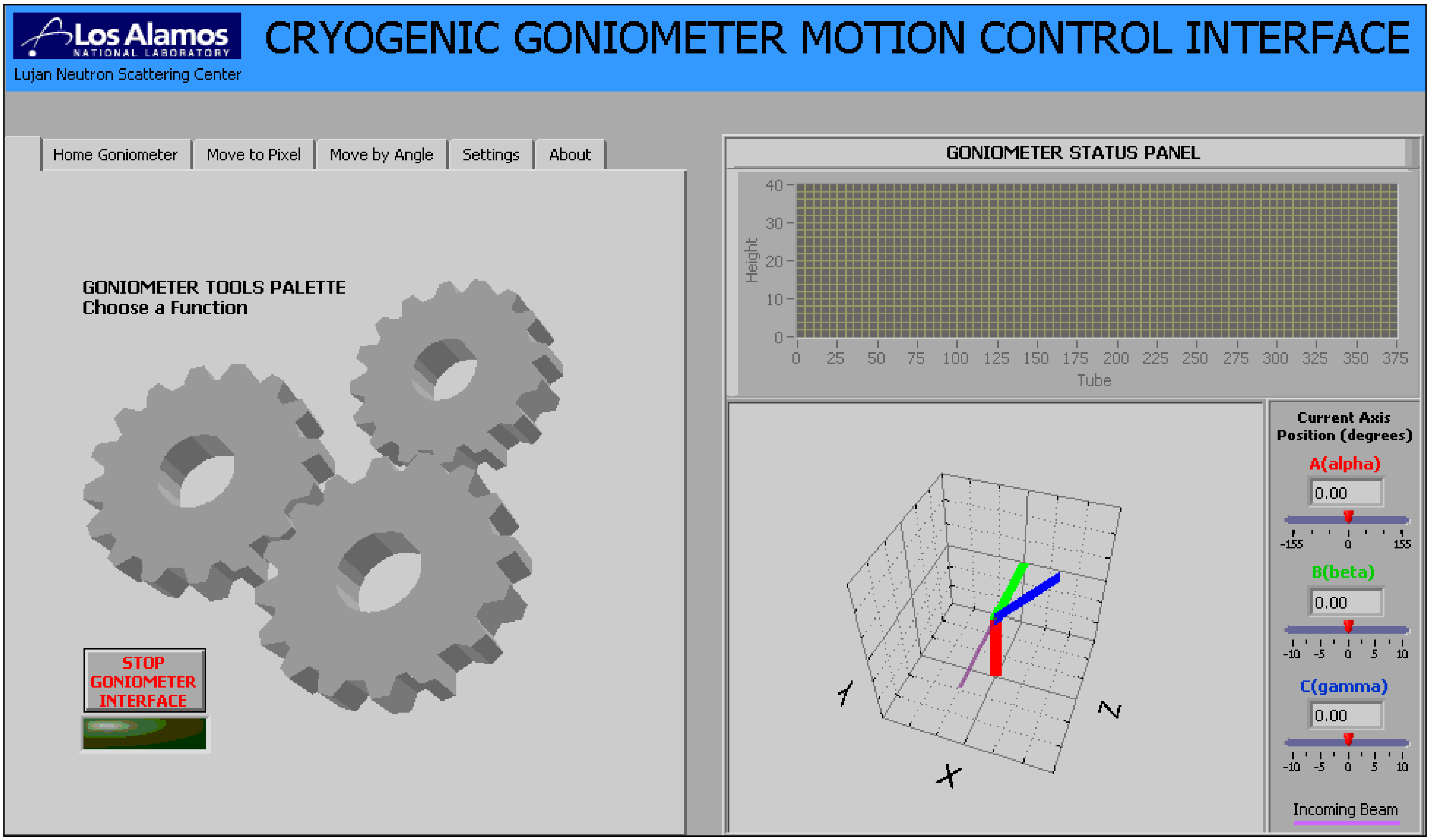
Appendix D: Complete LabVIEW Documentation of GoniometerMotionControl.vi

Does not include the low-level VIs from the Viewpoint Systems Motion Library

GoniometerMotionControl.vi

Connector Pane

Front Panel



Controls and Indicators

TF **stop**

Tools Palette

3D Curve ActiveX container that holds the 3D graph control.

TF **OK Button**

TF **OK Button 2**

TF **OK Button 3**

TF **OK Button 4**

TF **OK Button 5**

TF **OK Button 6**

Goniometer Properties

- DBL **A Range**
- DBL **B Range**
- DBL **C Range**
- DBL **Bracket Angle**

TF **OK Button 7**

TF **OK Button 10**

TF **OK Button 9**

TF **OK Button 11**

DBL **A (alpha)**

- **B (beta)**
- **C (gamma)**
- **Diffraction**
  - **CurrentTube**
  - **CurrentHeight**
  - **TargetTube**
  - **TargetHeight**
- **Pharos Detector Bank**
- **Numeric**
- **Numeric 2**
- **Numeric 3**
- **Slide**
- **Slide 2**
- **Slide 3**
- **alpha**
- **gamma**
- **beta**
- **Residual**
- **Boolean**

abc **Message**

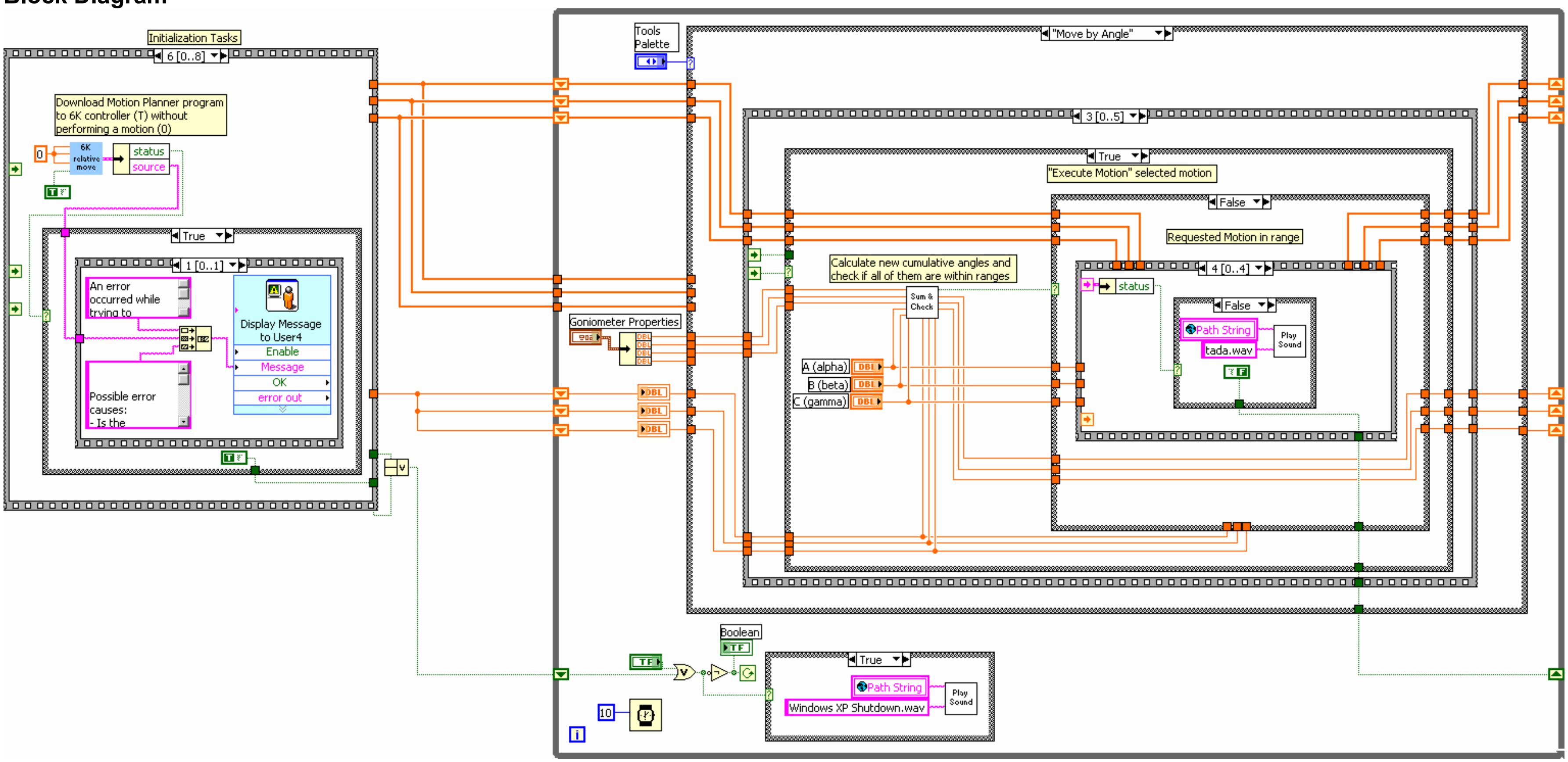
DBL **Slide 4**

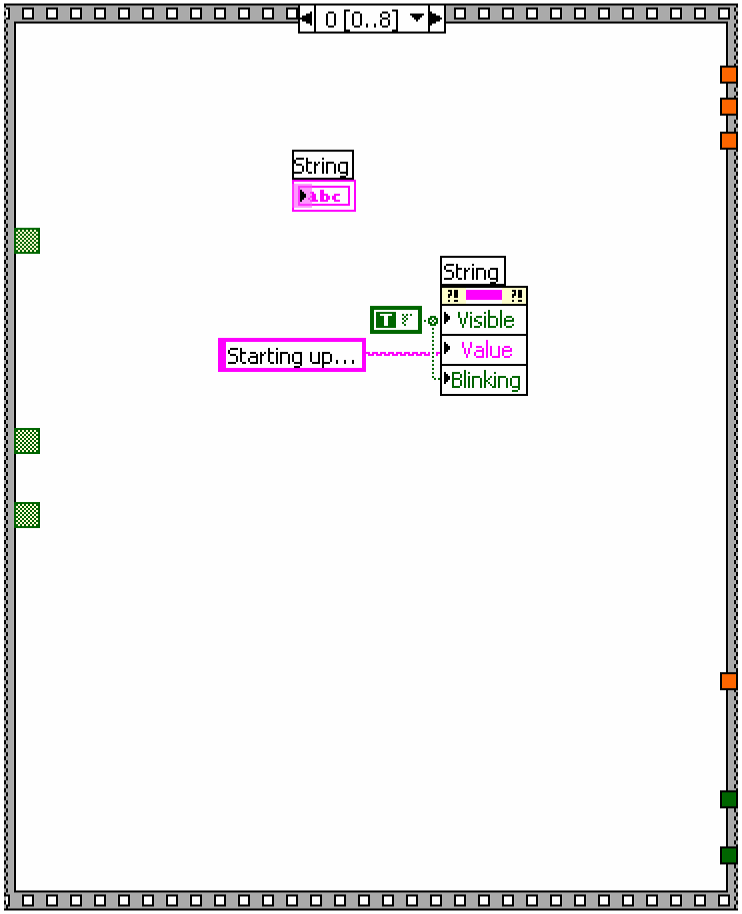
DBL **Slide 6**

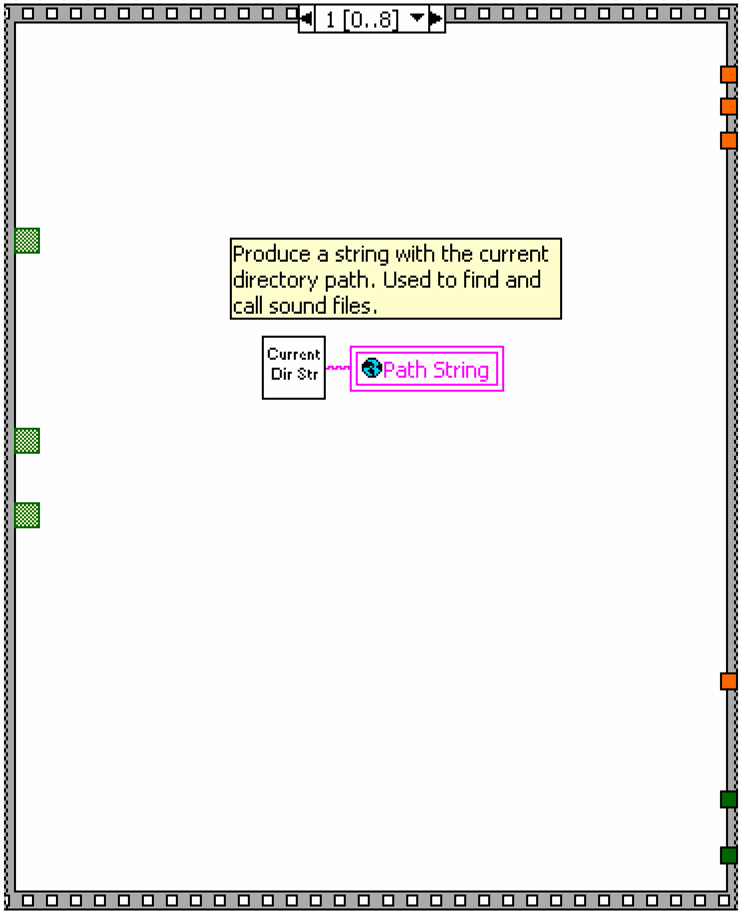
TF **Homing Motion in Progress**

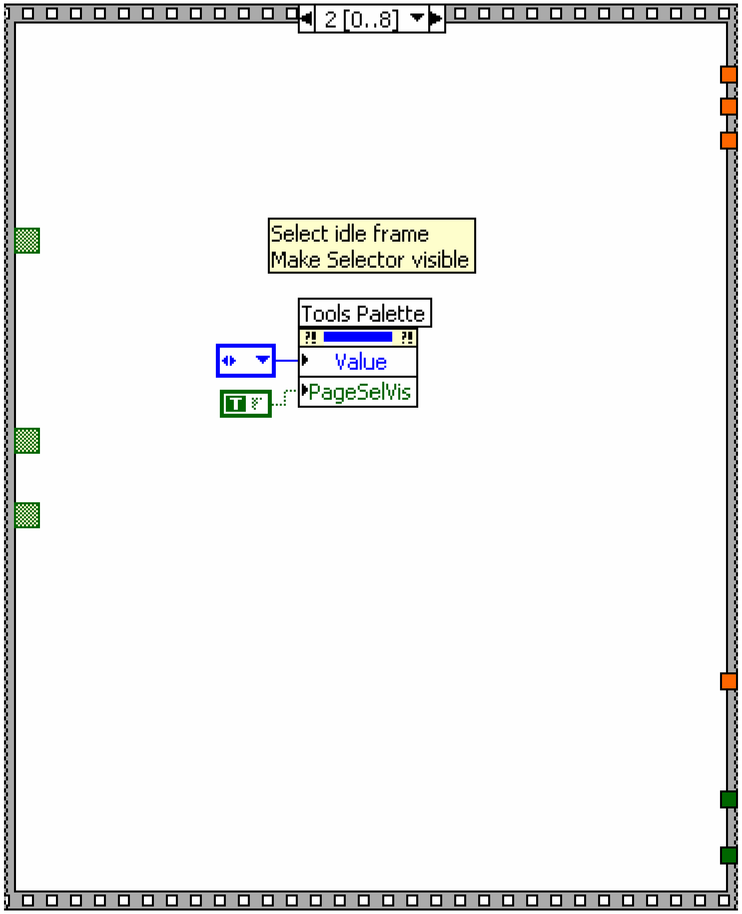
abc **String**

Block Diagram

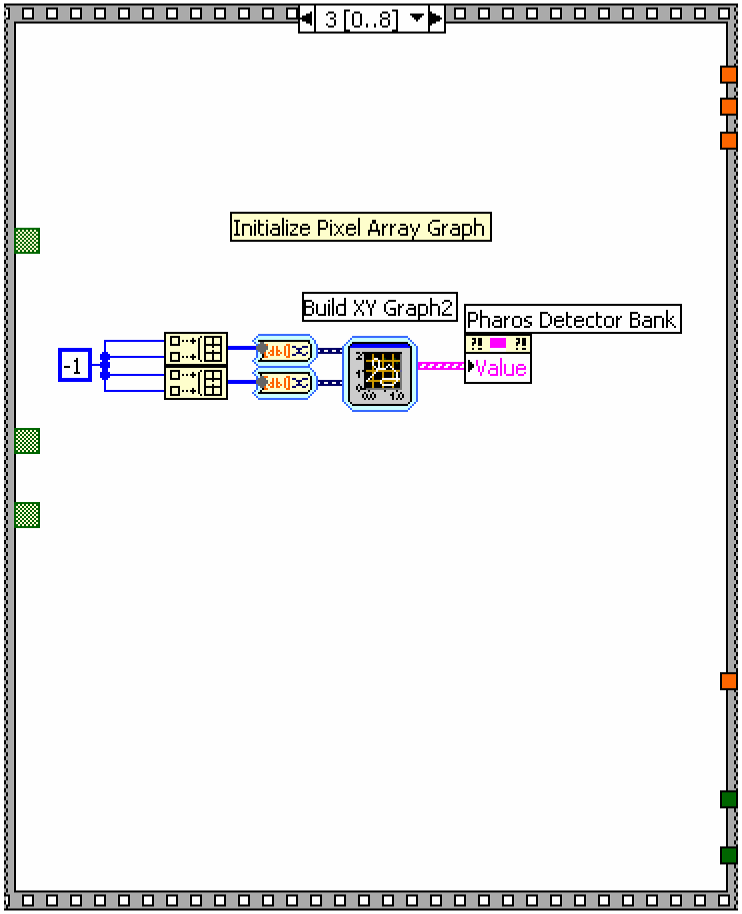


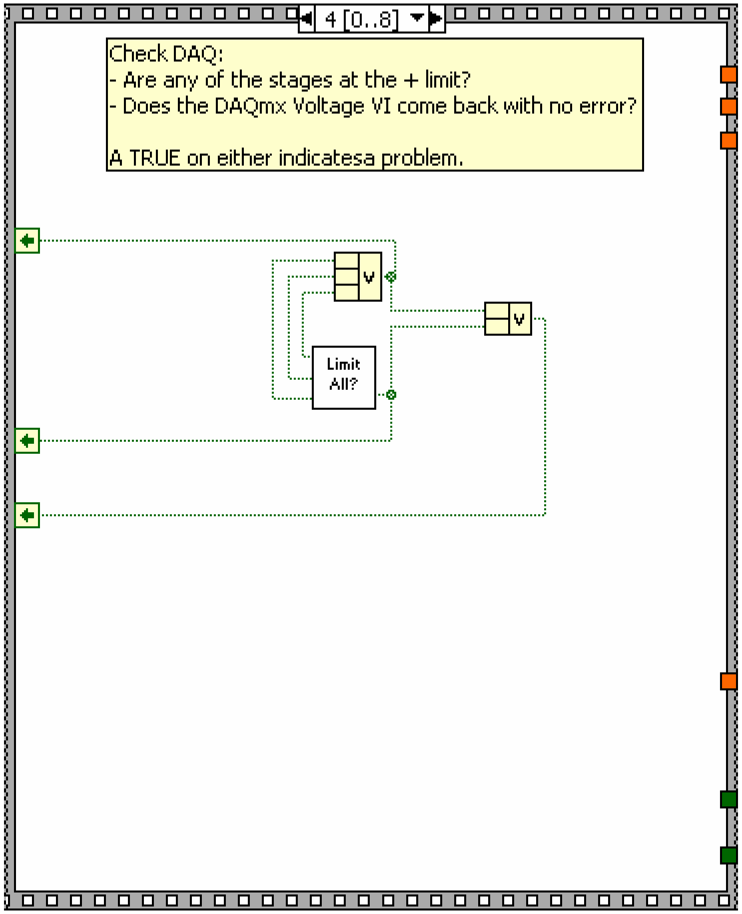

0 [0..8]
String
abc
String
Visible
Value
Blinking
Starting up...

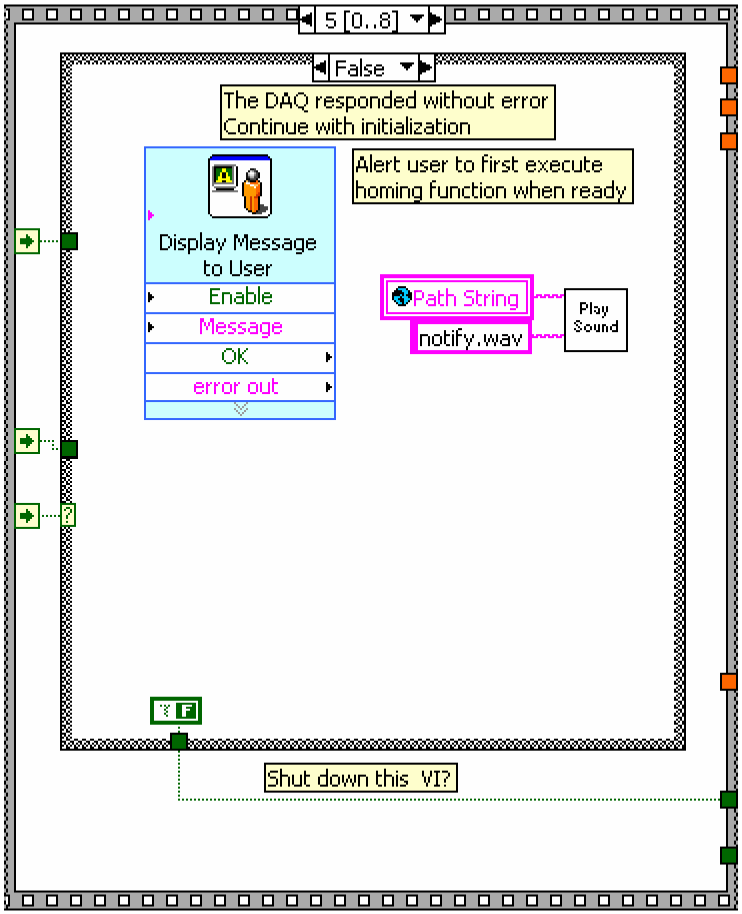

1 [0..8]
Produce a string with the current directory path. Used to find and call sound files.
Current Dir Str
Path String

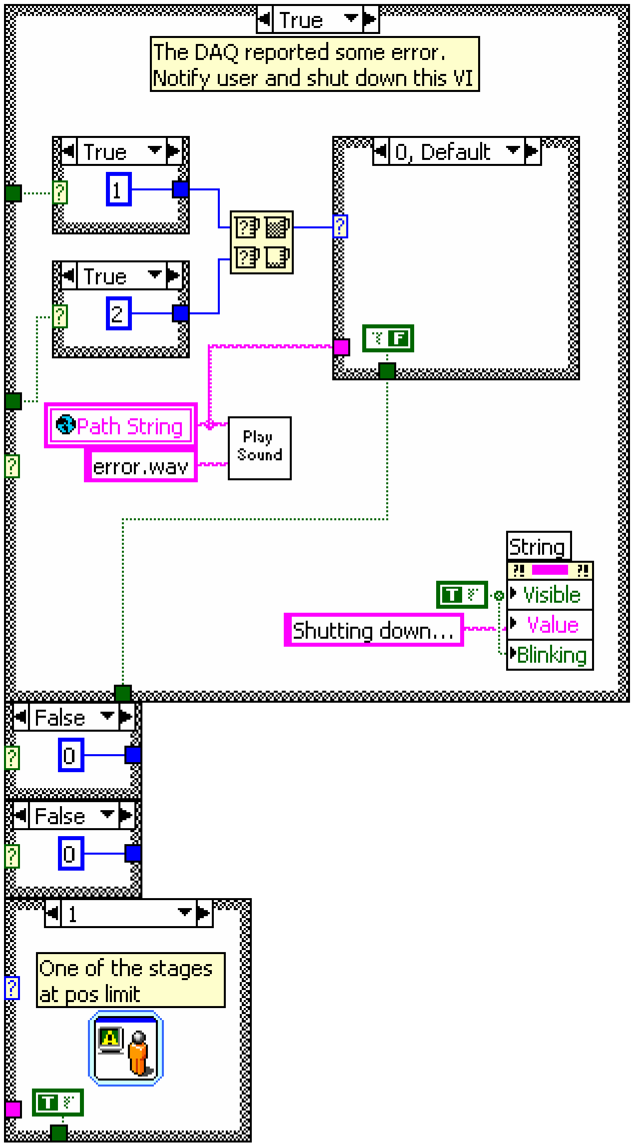


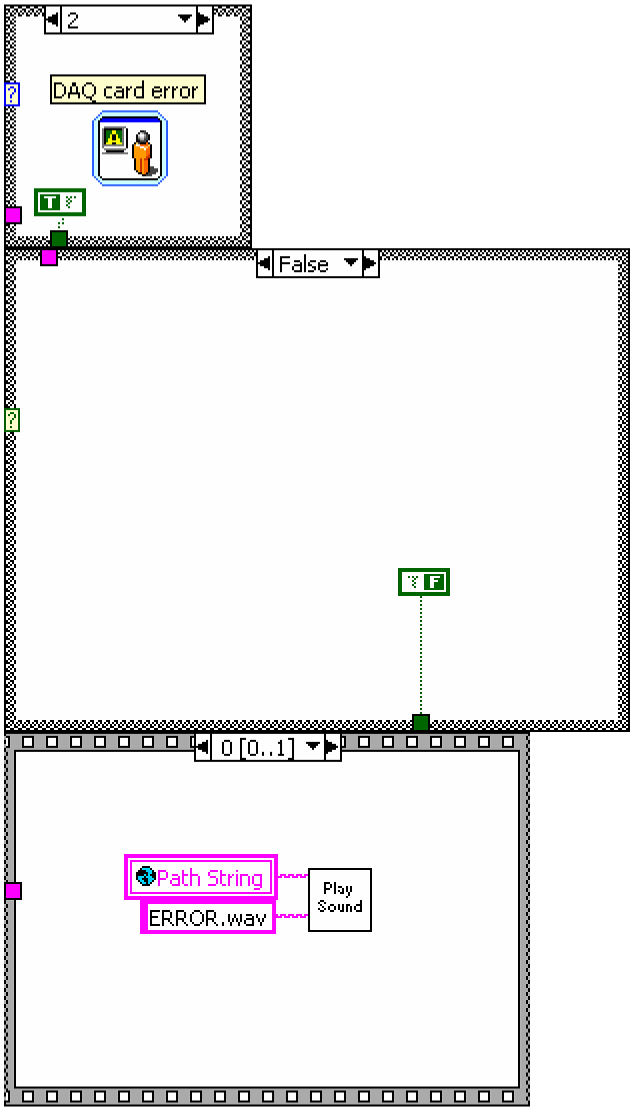
2 [0..8]
Select idle frame
Make Selector visible
Tools Palette
Value
PageSelVis

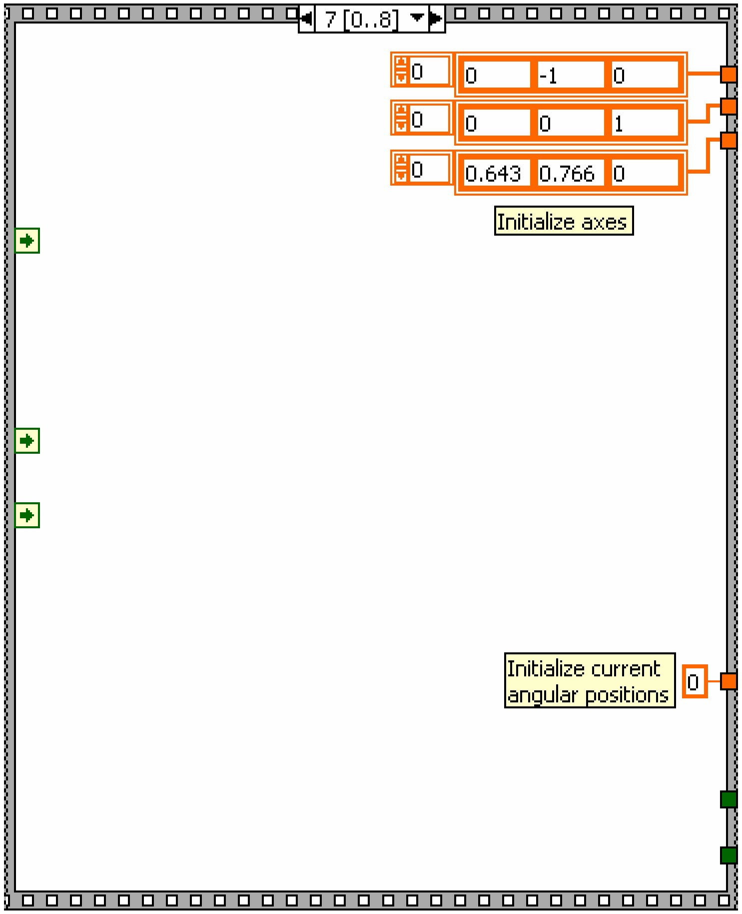

3 [0..8]
Initialize Pixel Array Graph
Build XY Graph2
Pharos Detector Bank
-1
Value

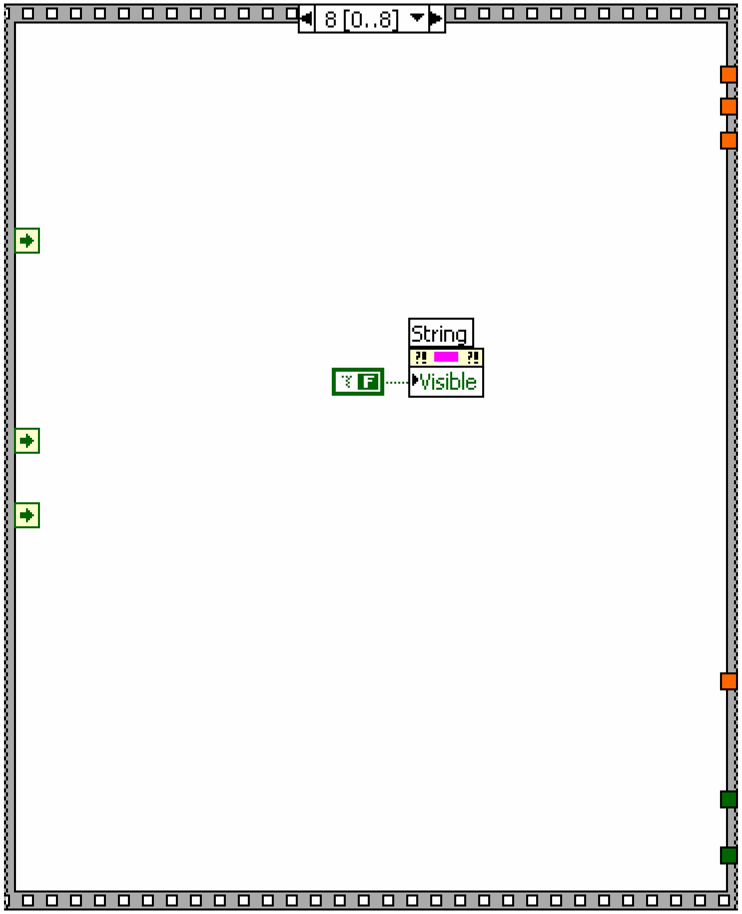

4 [0..8]
Check DAQ:
- Are any of the stages at the + limit?
- Does the DAQmx Voltage VI come back with no error?
A TRUE on either indicatesa problem.
Limit All?

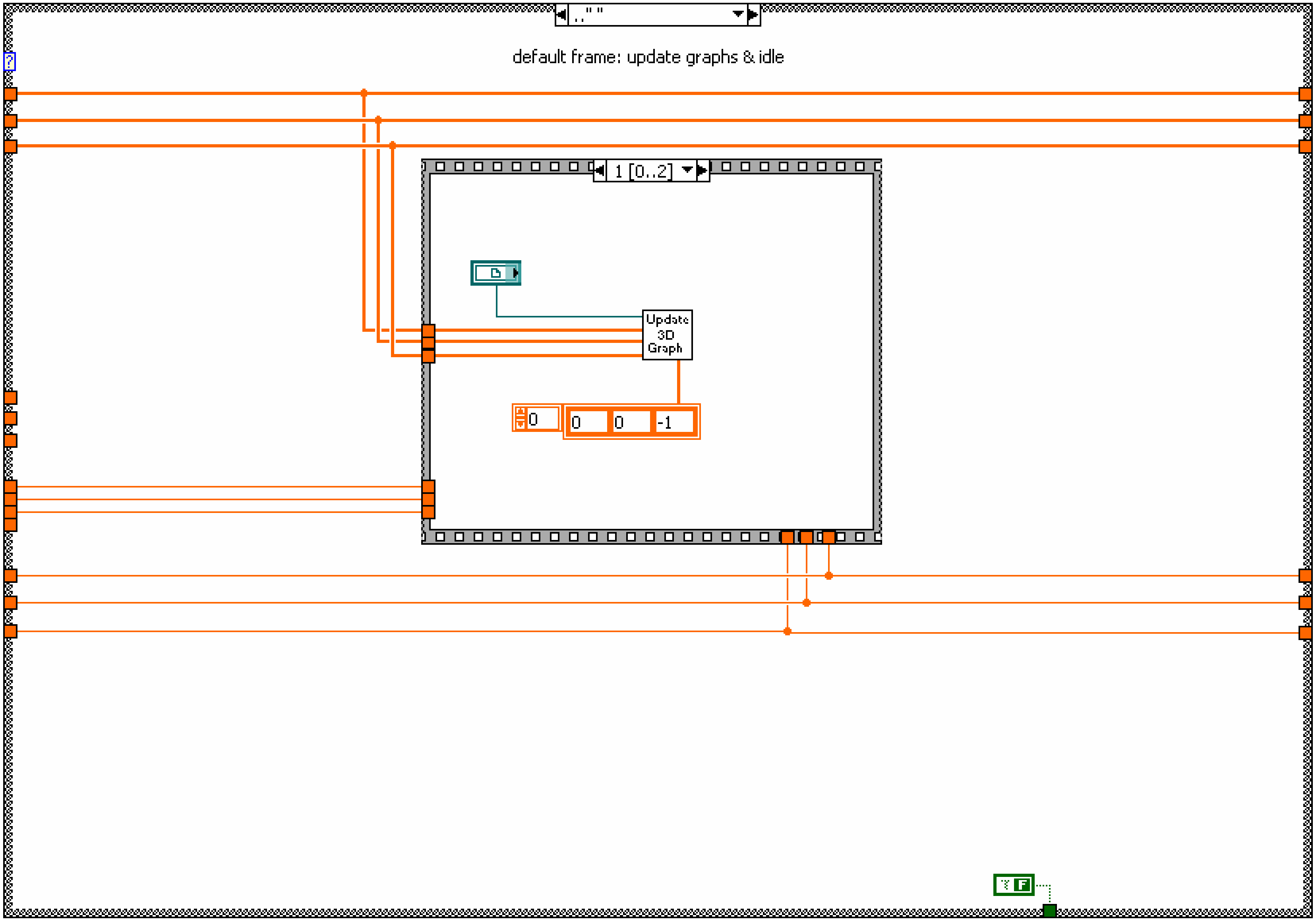

5 [0..8]
False
The DAQ responded without error
Continue with initialization
Display Message to User
Enable
Message
OK
error out
Alert user to first execute
homing function when ready
Path String
notify.wav
Play Sound
Shut down this VI?


True
The DAQ reported some error.
Notify user and shut down this VI
True
1
True
2
0, Default
Path String
error.wav
Play Sound
String
Visible
Value
Blinking
Shutting down...
False
0
False
0
1
One of the stages
at pos limit

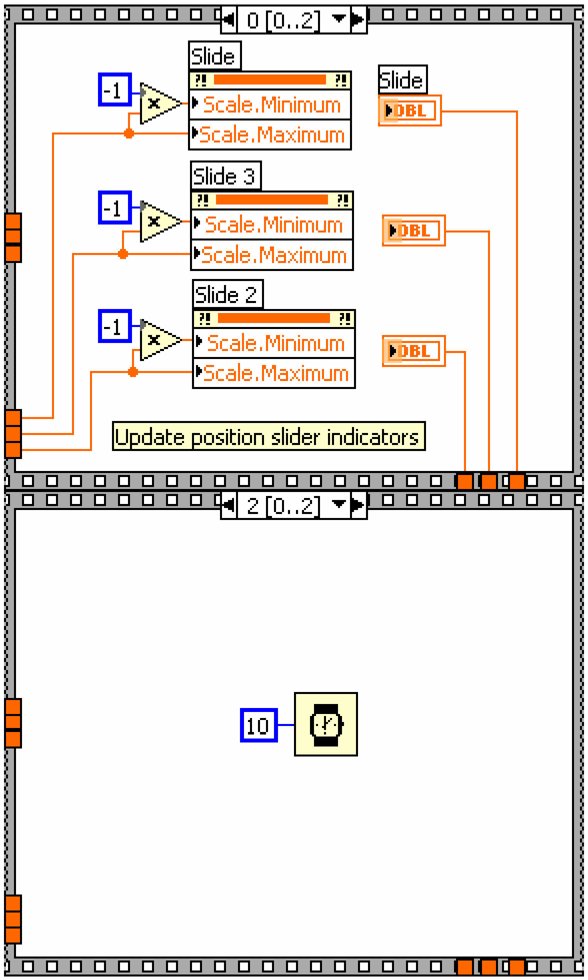

2
DAQ card error
False
0 [0..1]
Path String
ERROR.wav
Play Sound

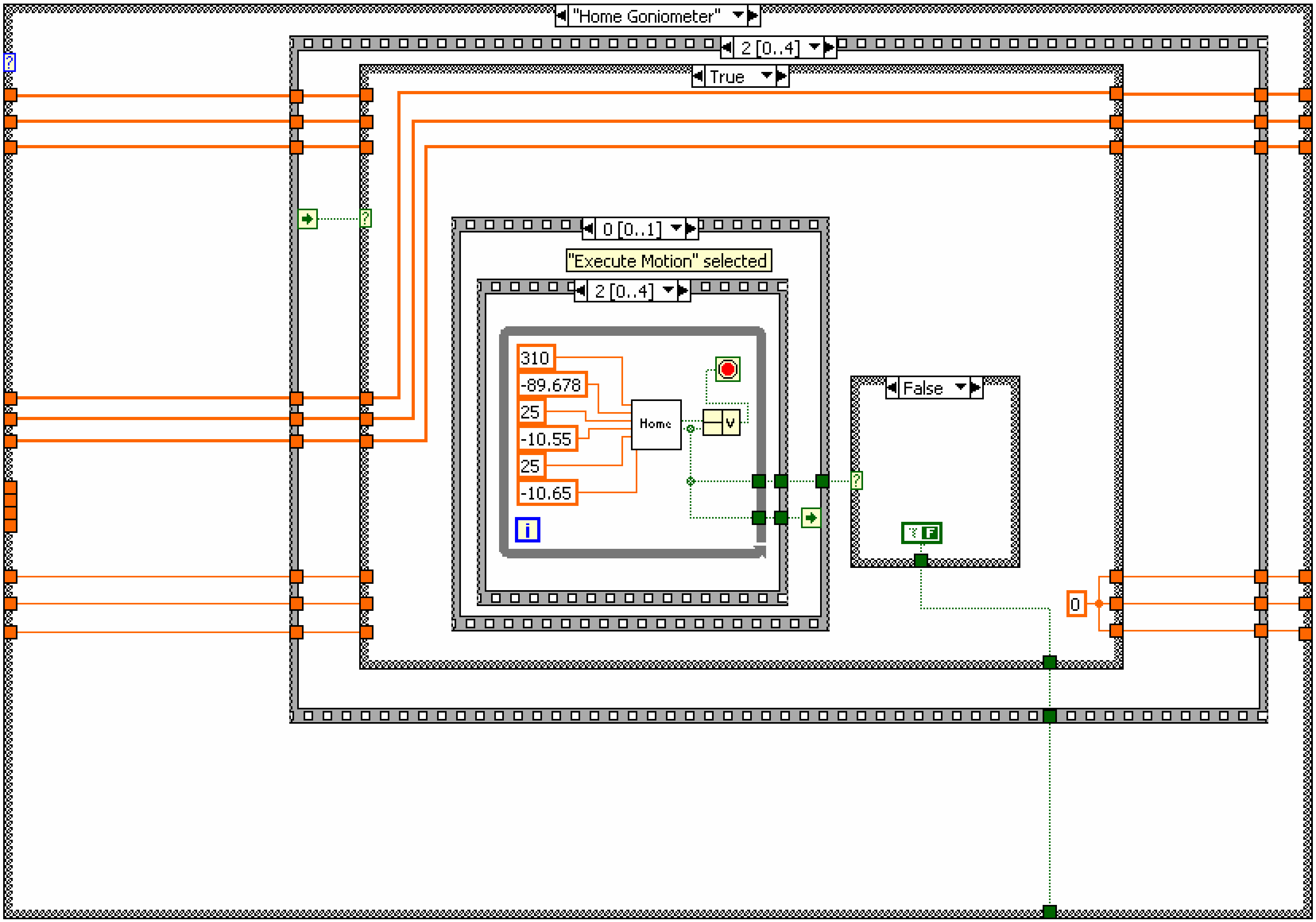

7 [0..8]
0
0
-1
0
0
0
0
1
0
0.643
0.766
0
Initialize axes
Initialize current angular positions
0

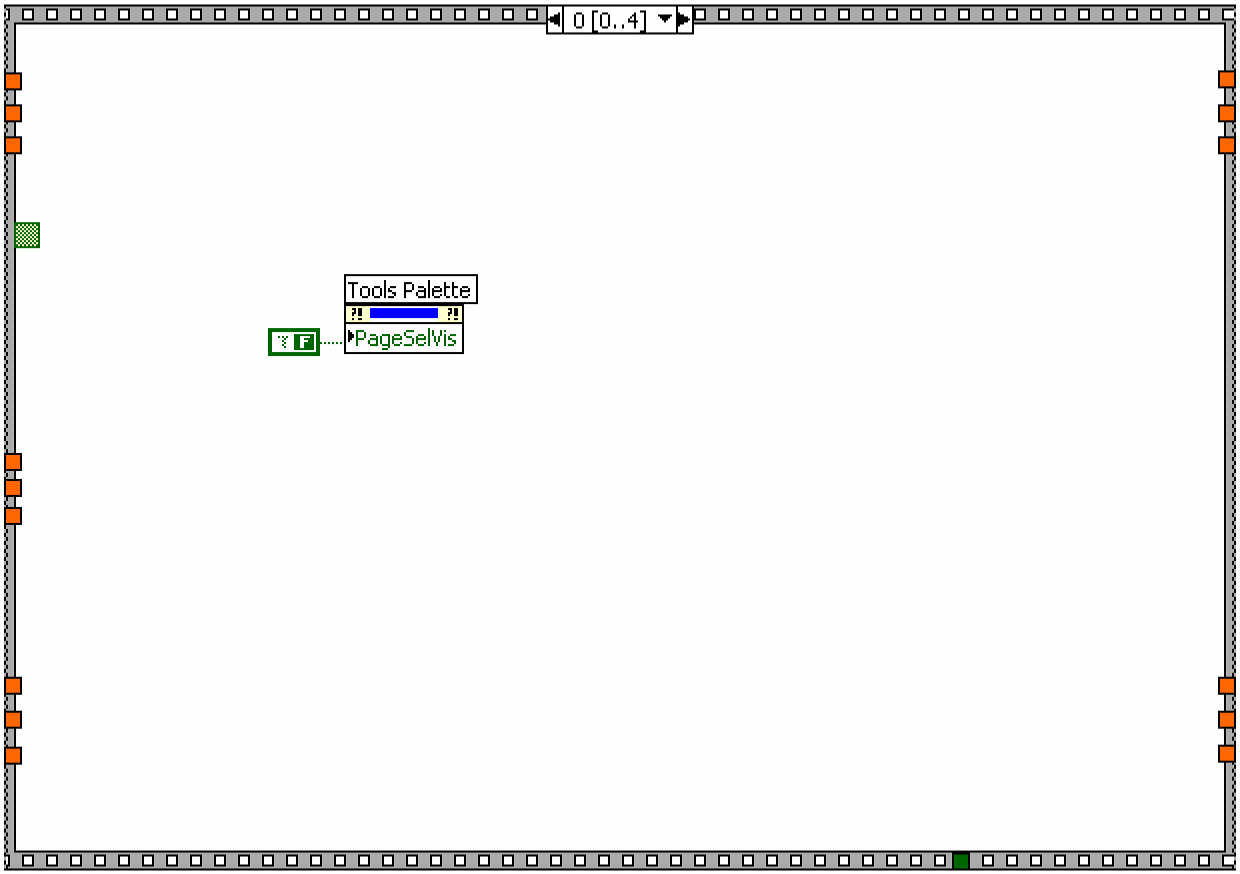

8 [0..8]
String
Visible

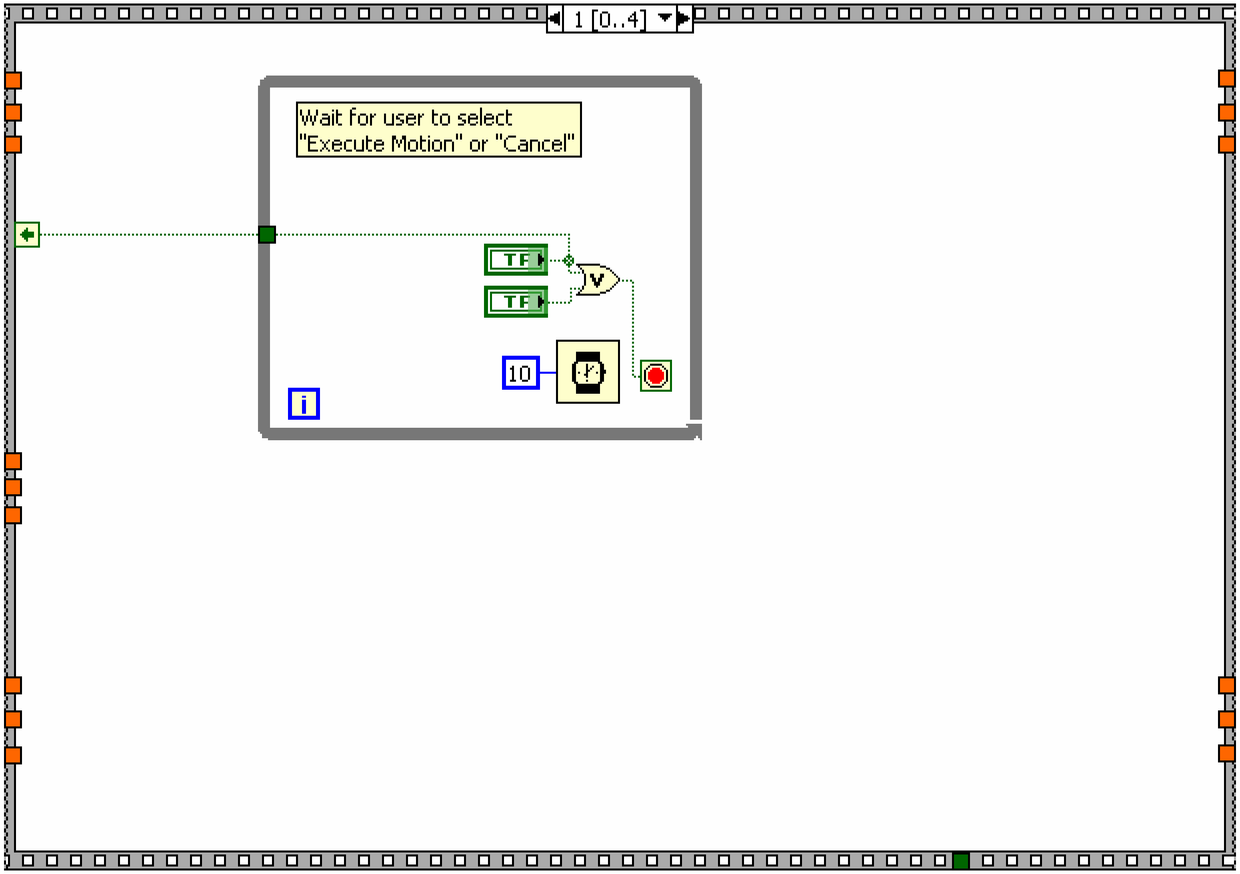


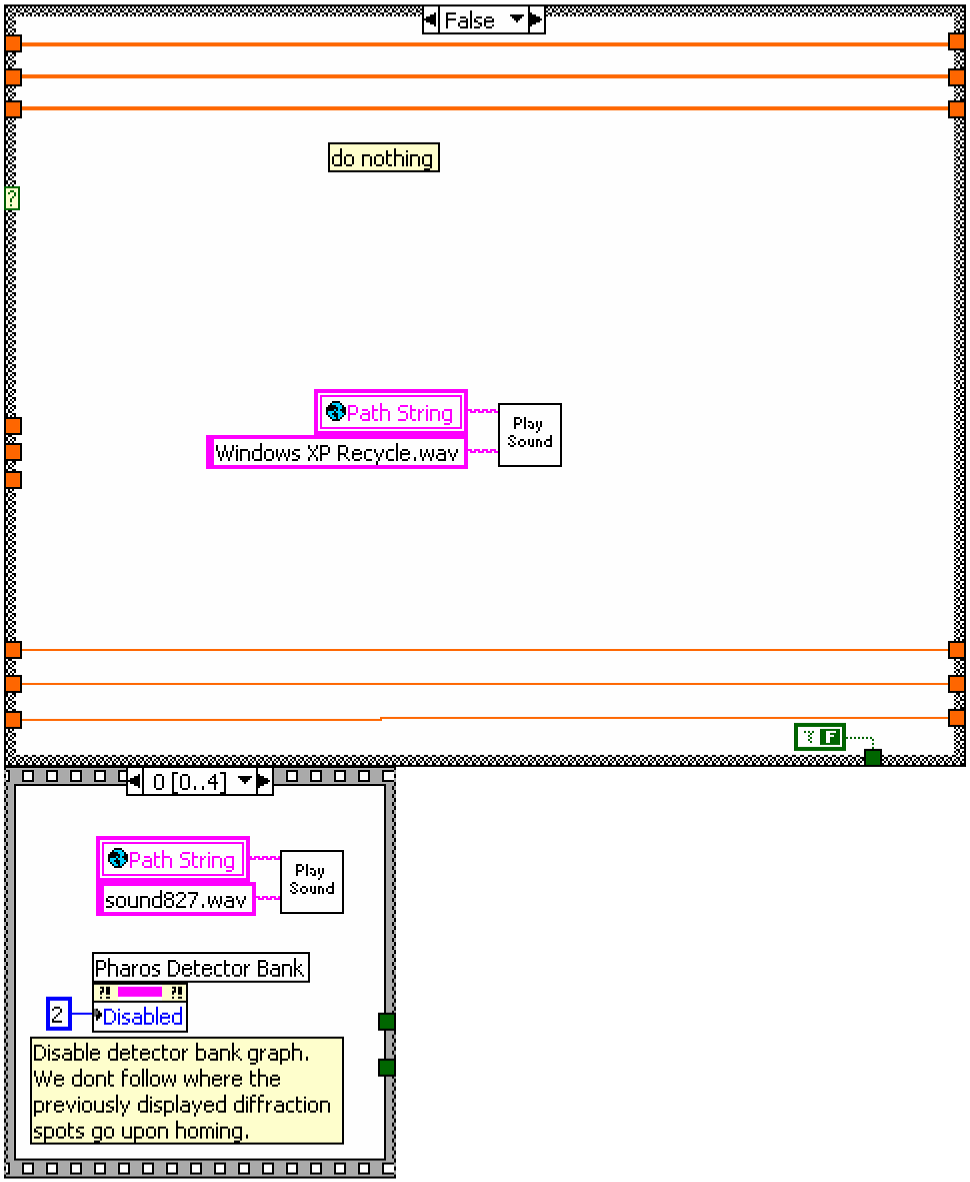
"..""
default frame: update graphs & idle
1 [0..2]
Update 3D Graph
0
0
0
-1
F

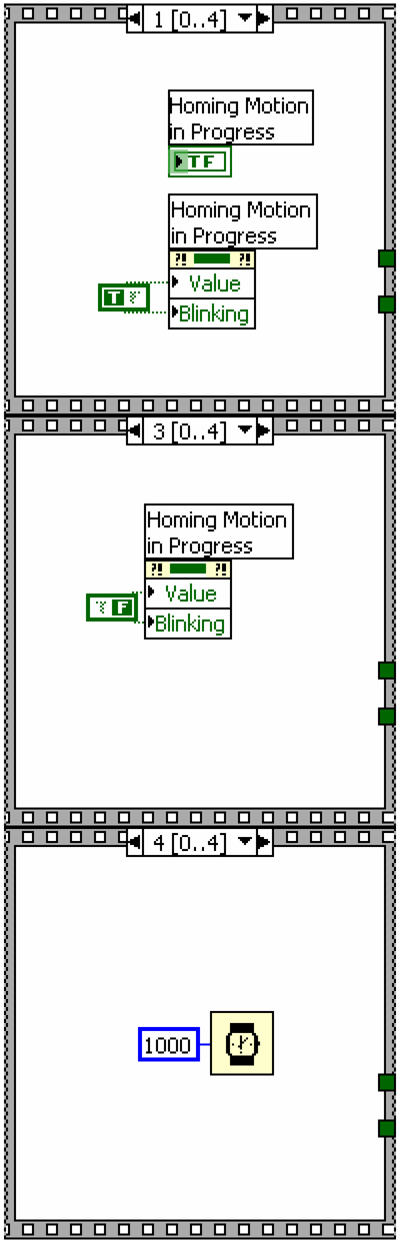

0 [0..2]
Slide
-1
Scale.Minimum
Scale.Maximum
Slide
DBL
Slide 3
-1
Scale.Minimum
Scale.Maximum
DBL
Slide 2
-1
Scale.Minimum
Scale.Maximum
DBL
Update position slider indicators
2 [0..2]
10

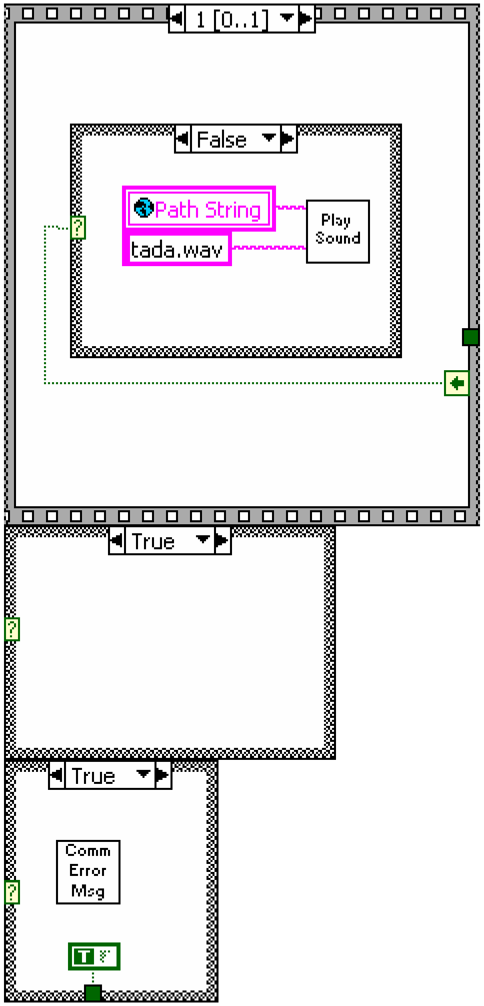

"Home Goniometer"
2 [0..4]
True
0 [0..1]
"Execute Motion" selected
2 [0..4]
310
-89.678
25
-10.55
25
-10.65
Home
False
0

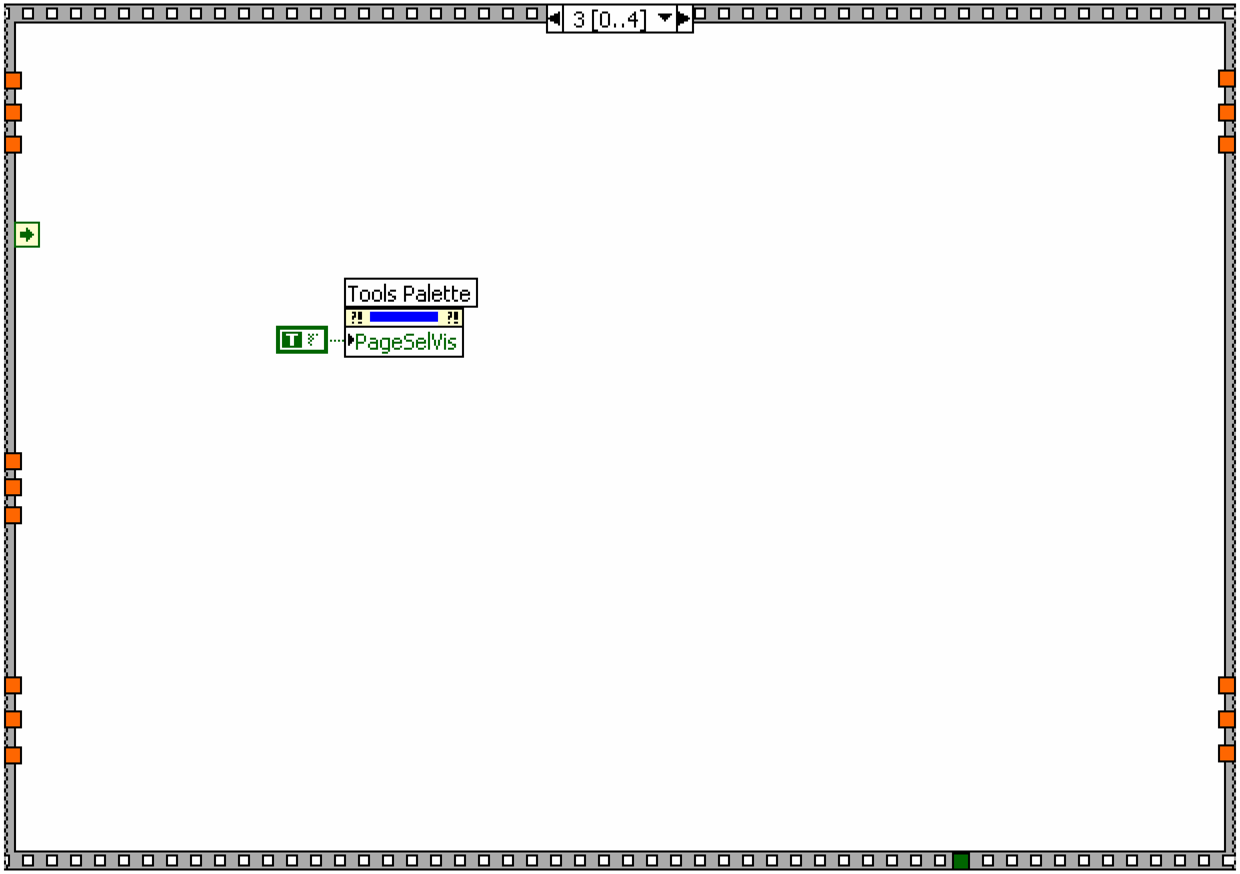

0 [0..4]
Tools Palette
PageSelVis
F

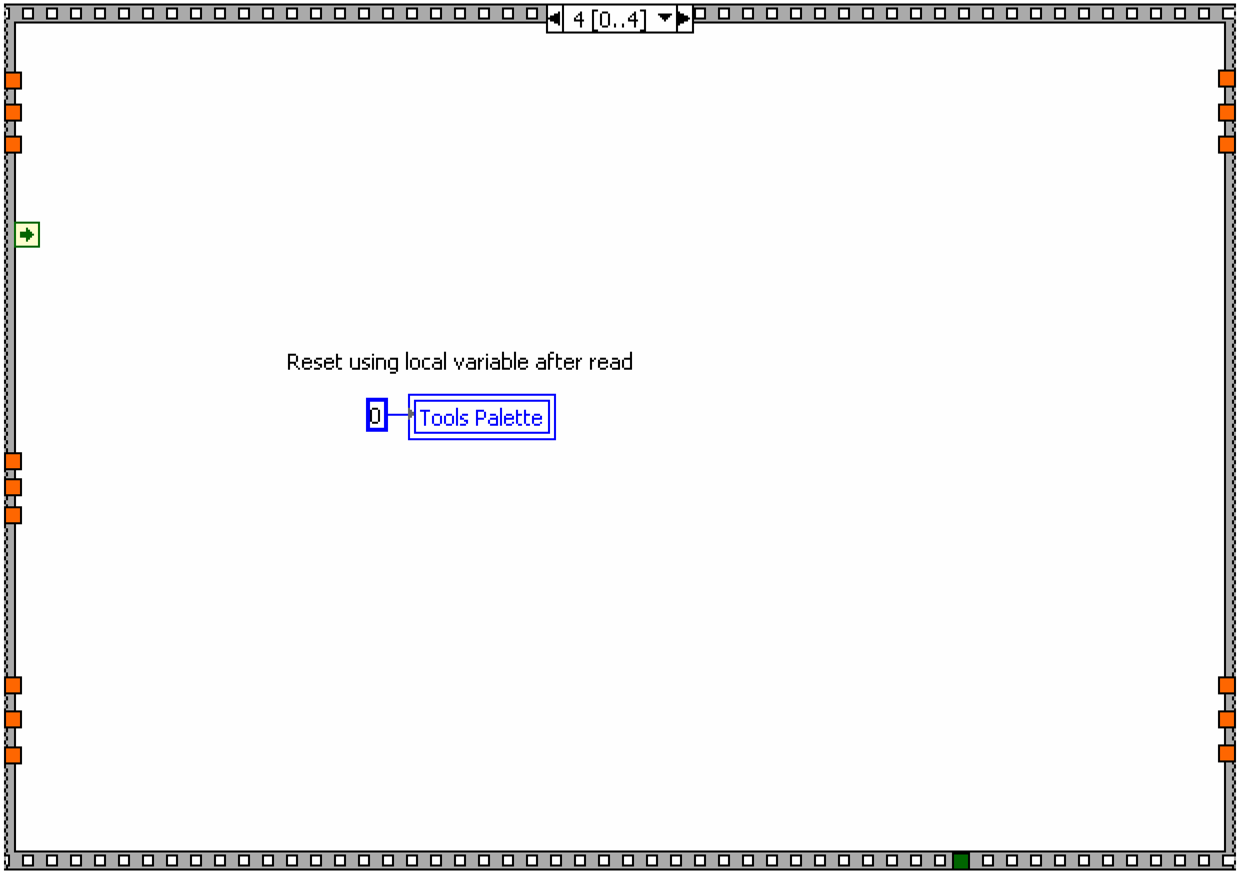

1 [0..4]
Wait for user to select
"Execute Motion" or "Cancel"
TF
TF
10
i

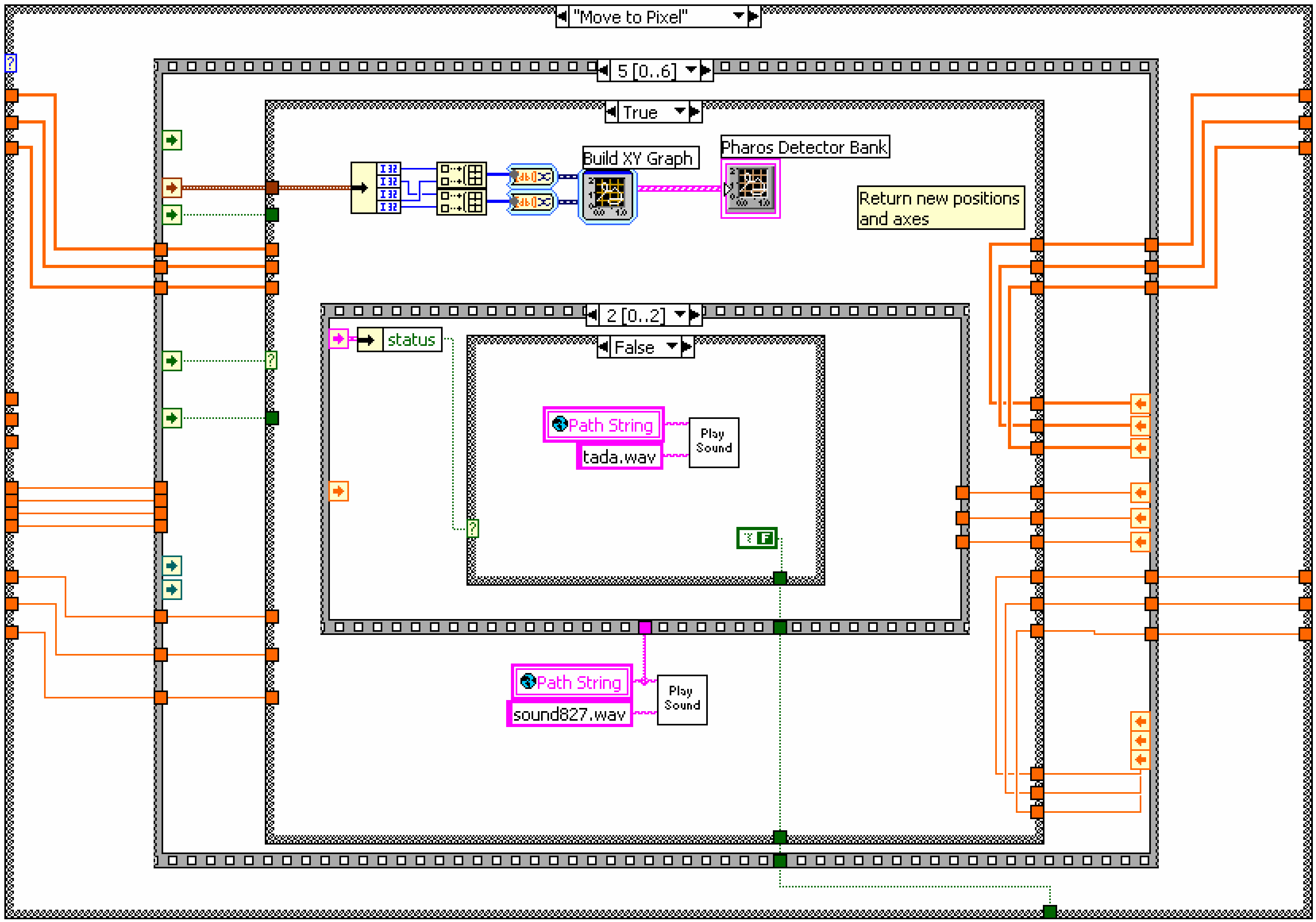

False
do nothing
Path String
Windows XP Recycle.wav
Play Sound
0 [0..4]
Path String
sound827.wav
Play Sound
Pharos Detector Bank
2
Disabled
Disable detector bank graph.
We dont follow where the
previously displayed diffraction
spots go upon homing.


1 [0..4]
Homing Motion in Progress
Homing Motion in Progress
Value
Blinking
3 [0..4]
Homing Motion in Progress
Value
Blinking
4 [0..4]
1000

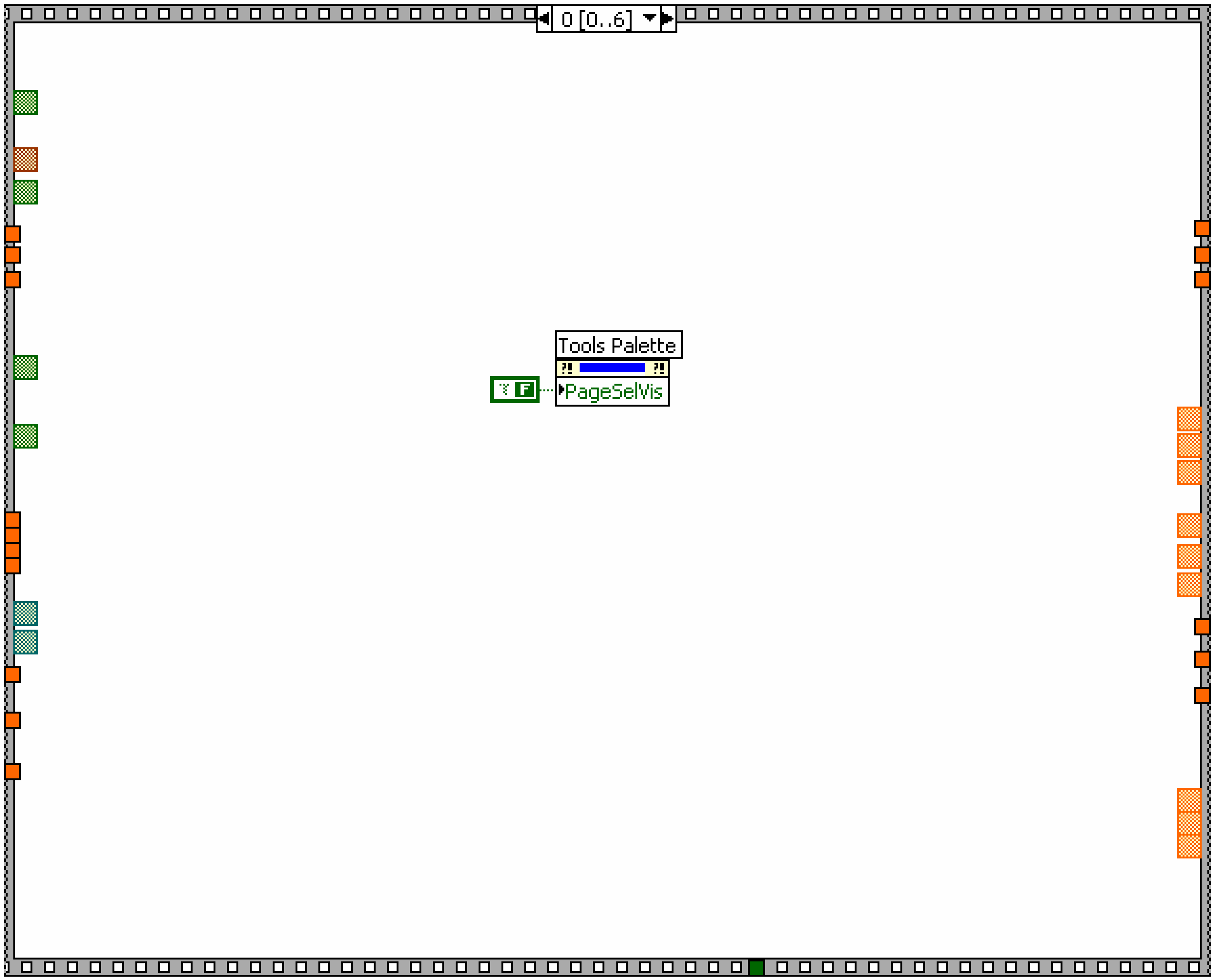

1 [0..1]
False
Path String
tada.wav
Play Sound
True
True
Comm Error Msg

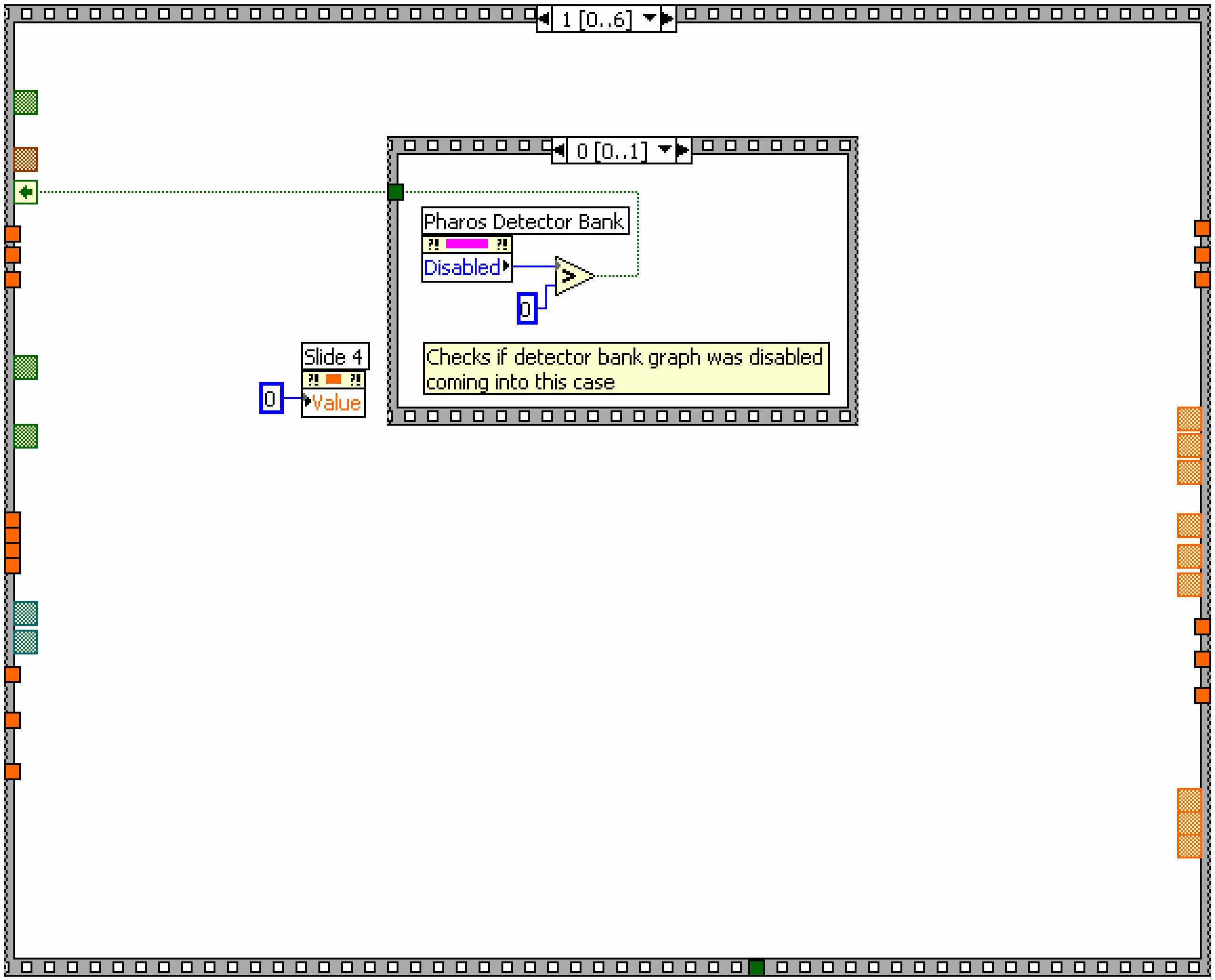

3 [0..4]
Tools Palette
PageSelVis

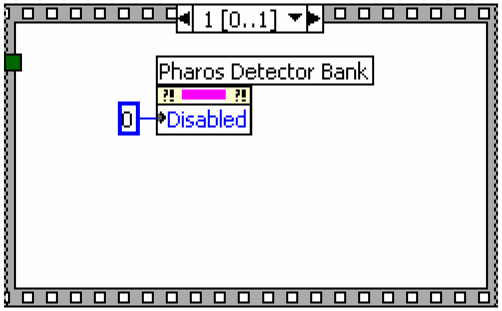

4 [0..4]
Reset using local variable after read
0
Tools Palette

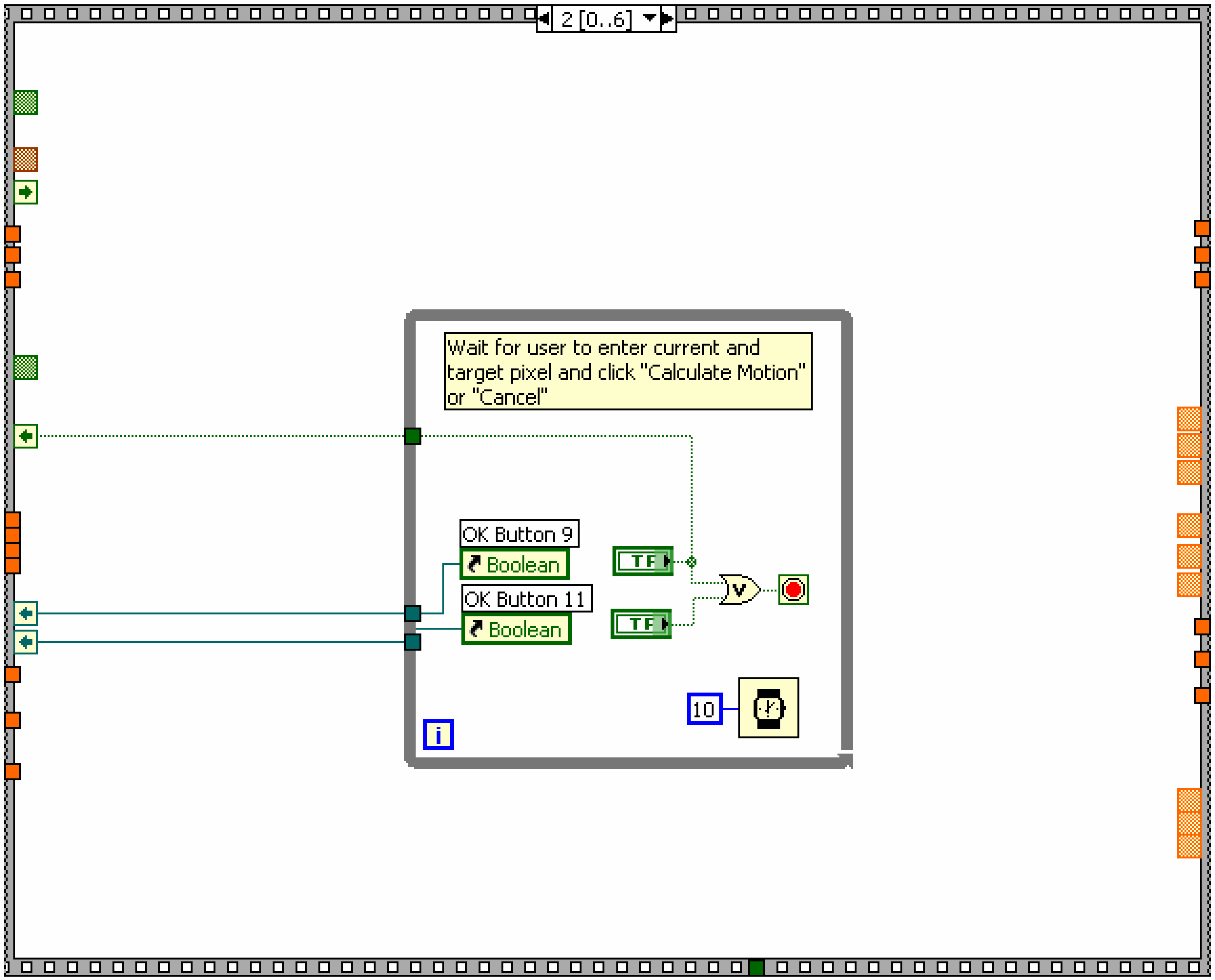


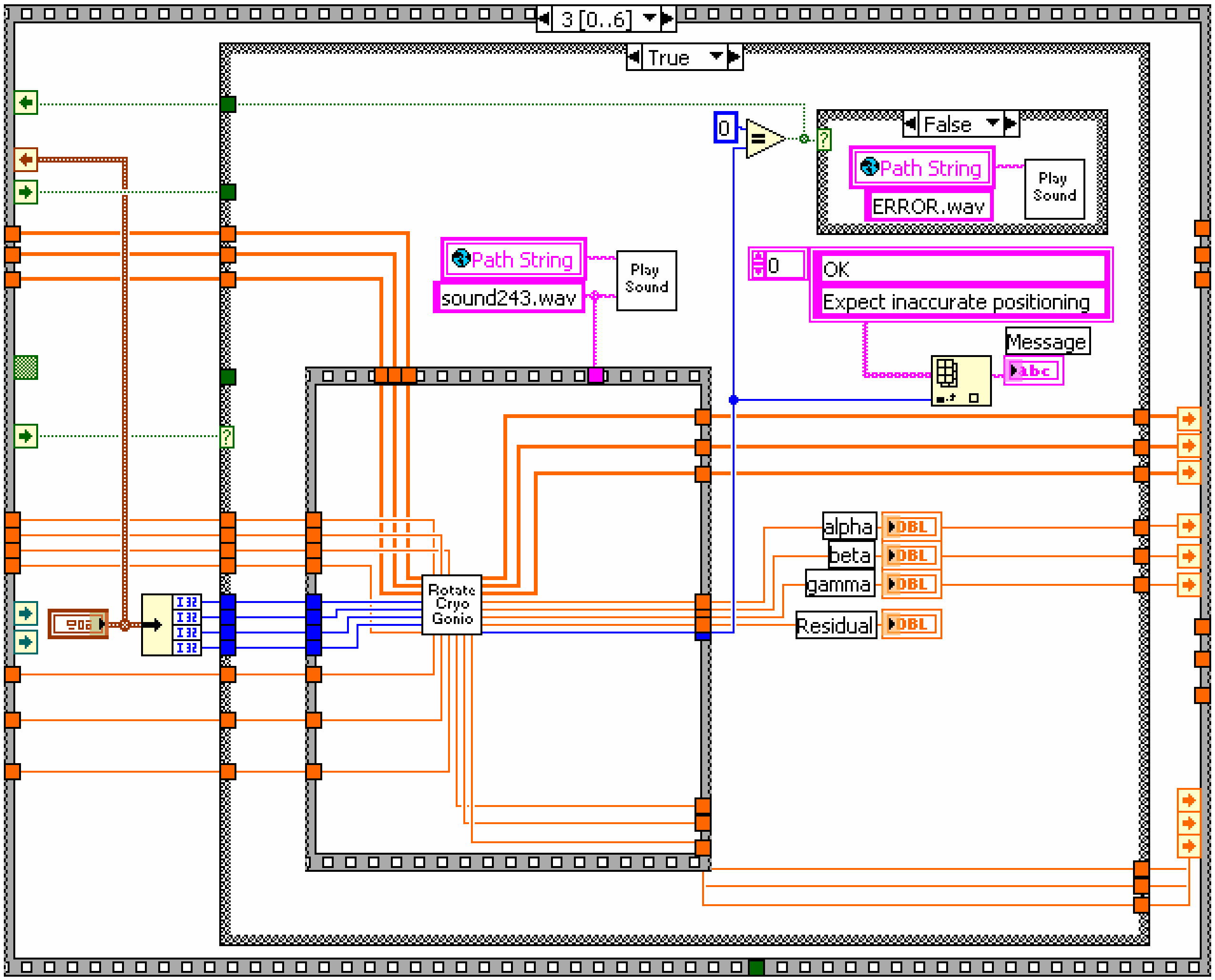
"Move to Pixel"
5 [0..6]
True
Build XY Graph
Pharos Detector Bank
Return new positions and axes
2 [0..2]
status
False
Path String
tada.wav
Play Sound
Path String
sound827.wav
Play Sound


0 [0..6]
Tools Palette
PageSelVis
F

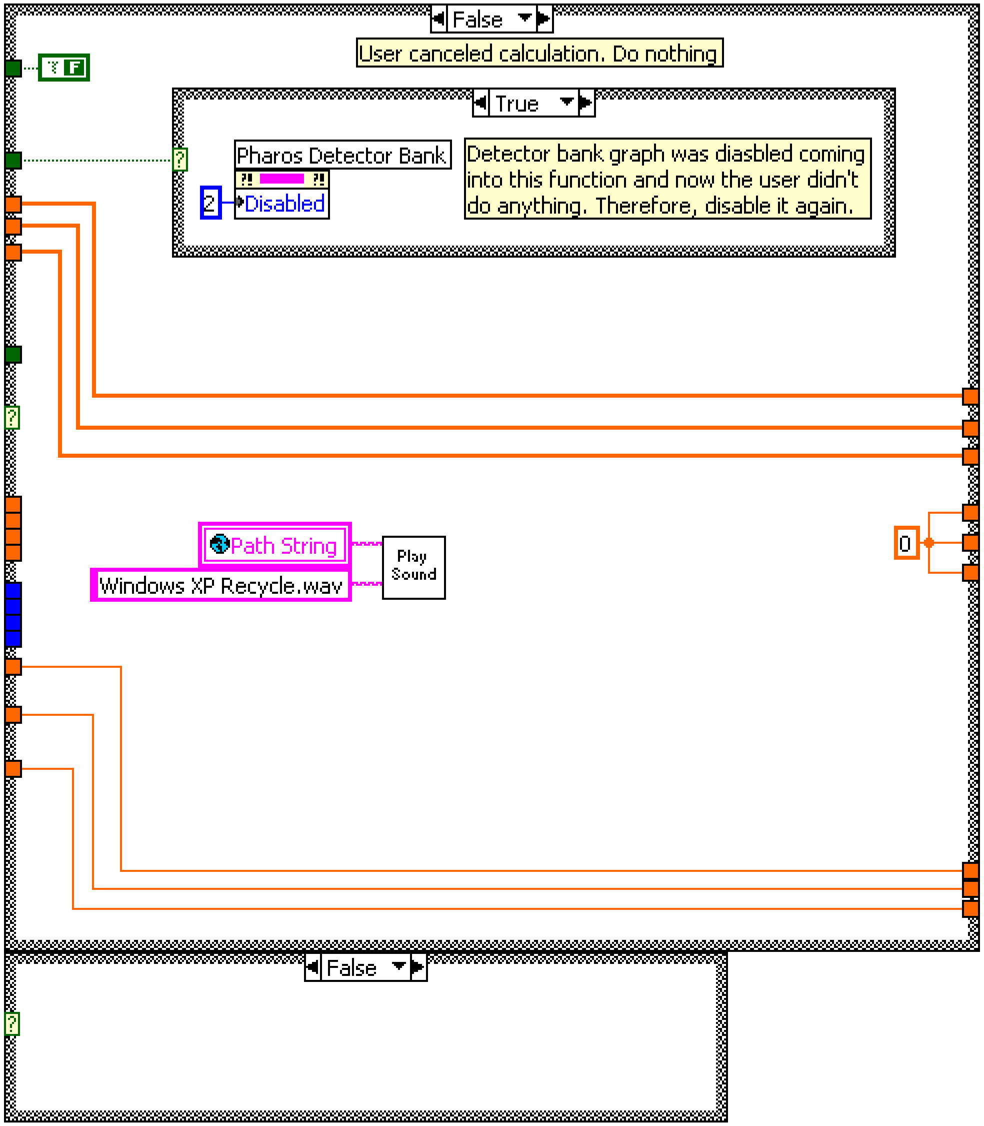

1 [0..6]
0 [0..1]
Pharos Detector Bank
Disabled
0
Slide 4
0
Value
Checks if detector bank graph was disabled coming into this case

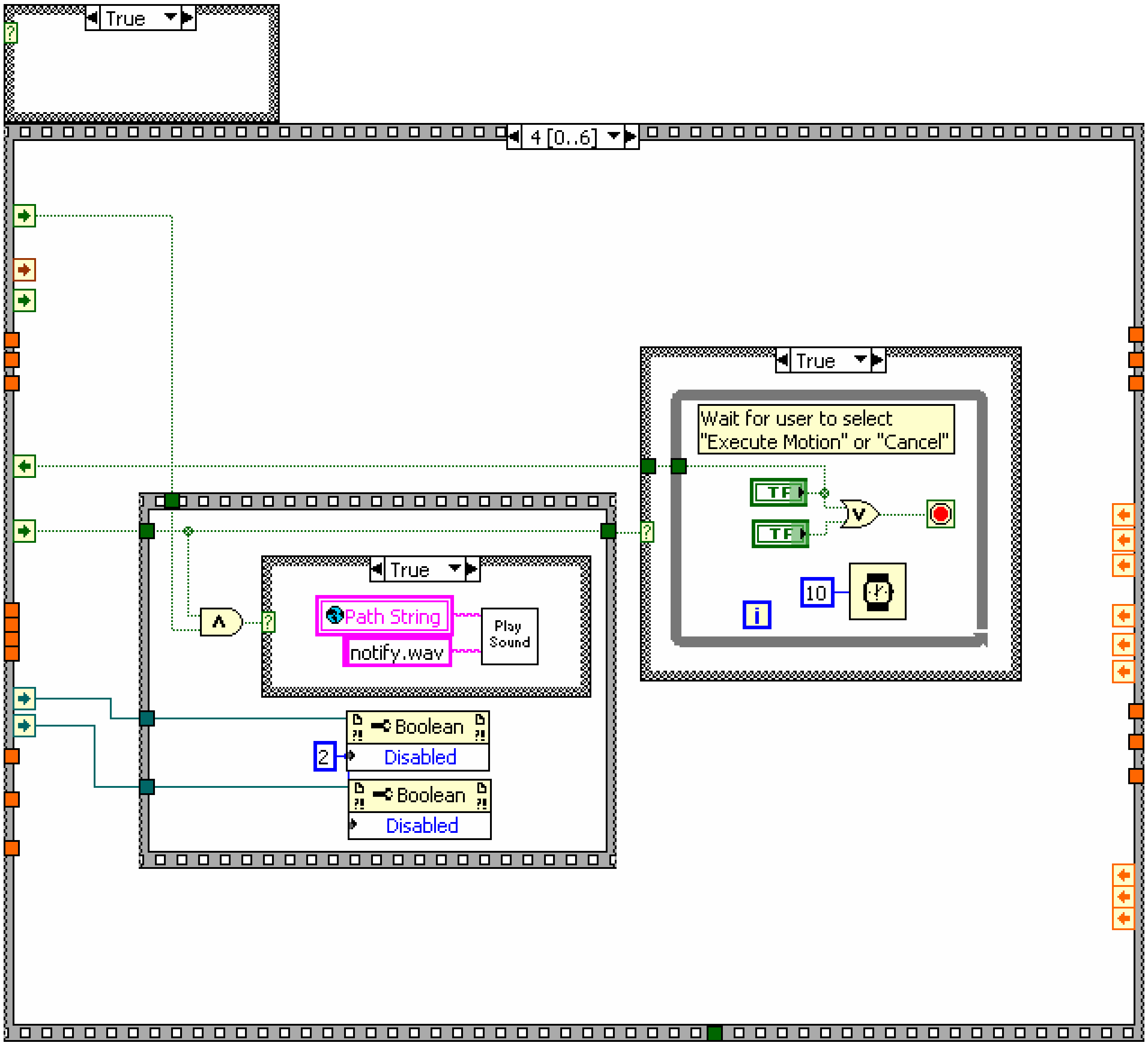

1 [0..1]
Pharos Detector Bank
0
Disabled

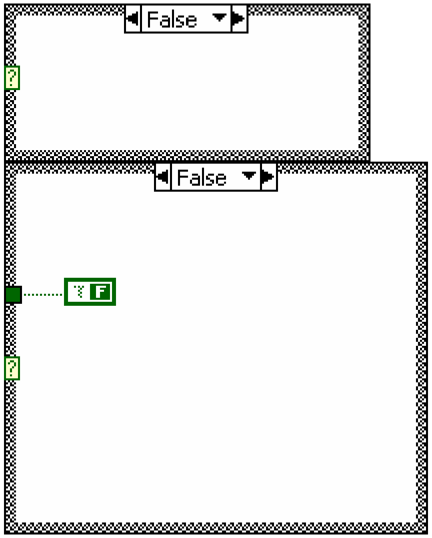

2 [0..6]
Wait for user to enter current and target pixel and click "Calculate Motion" or "Cancel"
OK Button 9
Boolean
OK Button 11
Boolean
TF
TF
10
i

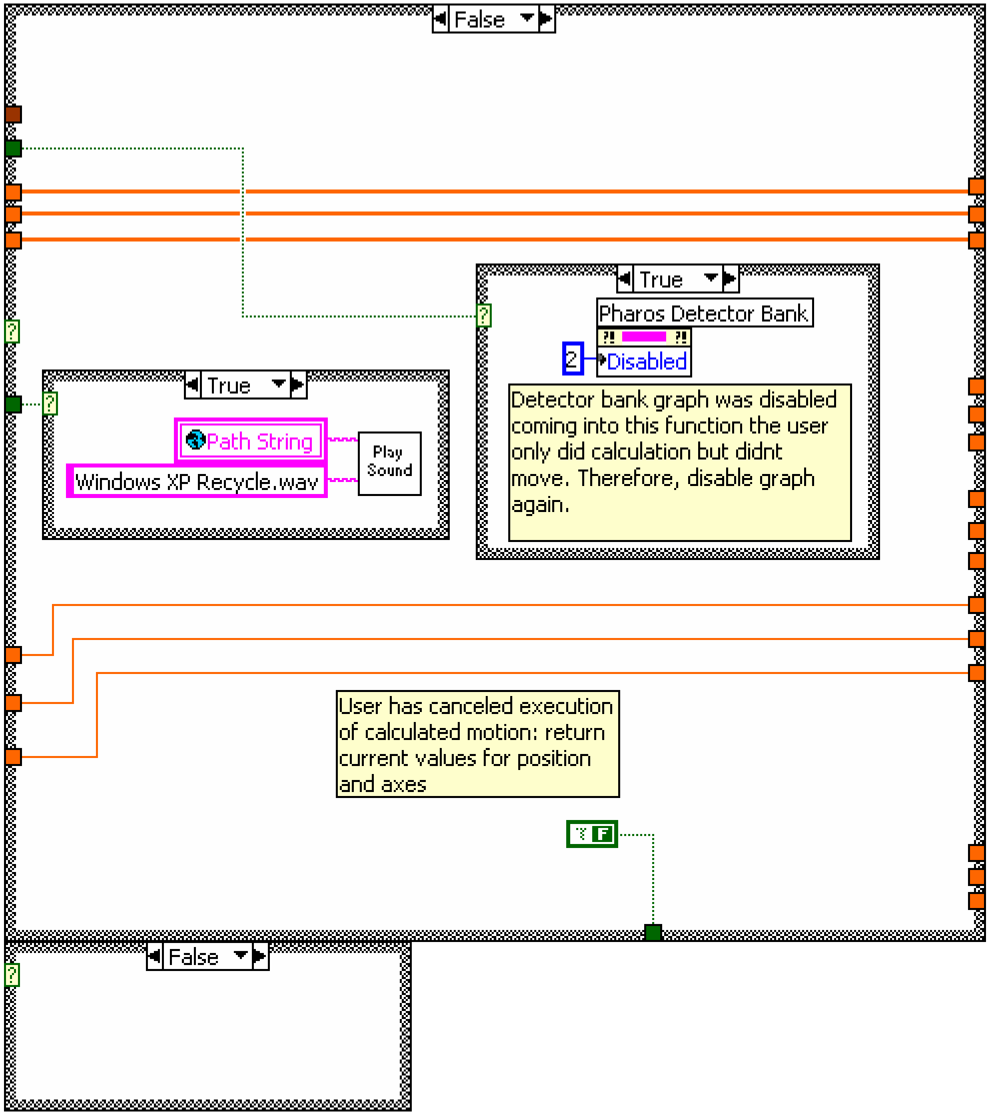


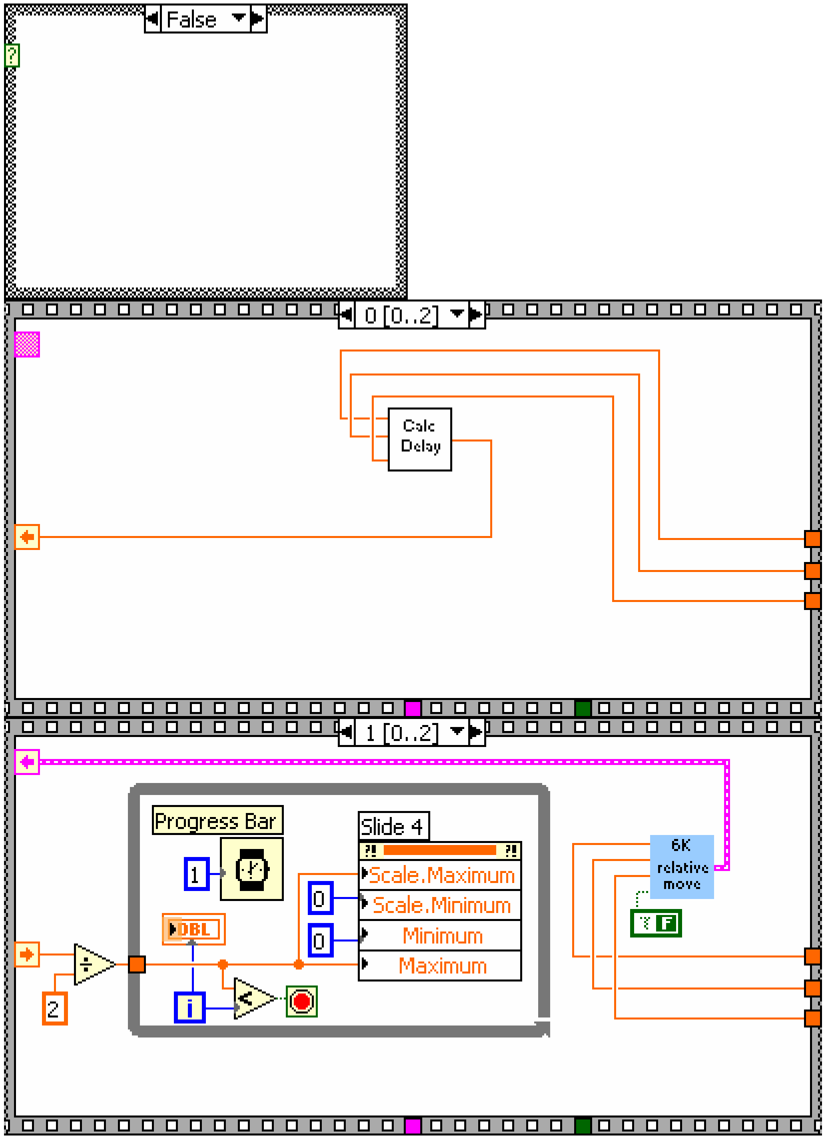
3 [0..6]
True
False
0
Path String
ERROR.wav
Play Sound
Path String
sound243.wav
Play Sound
0
OK
Expect inaccurate positioning
Message
abc
Rotate Cryo Gonio
alpha
beta
gamma
Residual
DBL
DBL
DBL
DBL

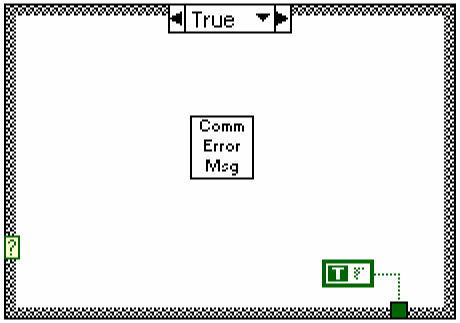

False
User canceled calculation. Do nothing
True
Pharos Detector Bank
Disabled
2
Detector bank graph was diasbled coming into this function and now the user didn't do anything. Therefore, disable it again.
Path String
Windows XP Recycle.wav
Play Sound
0
False

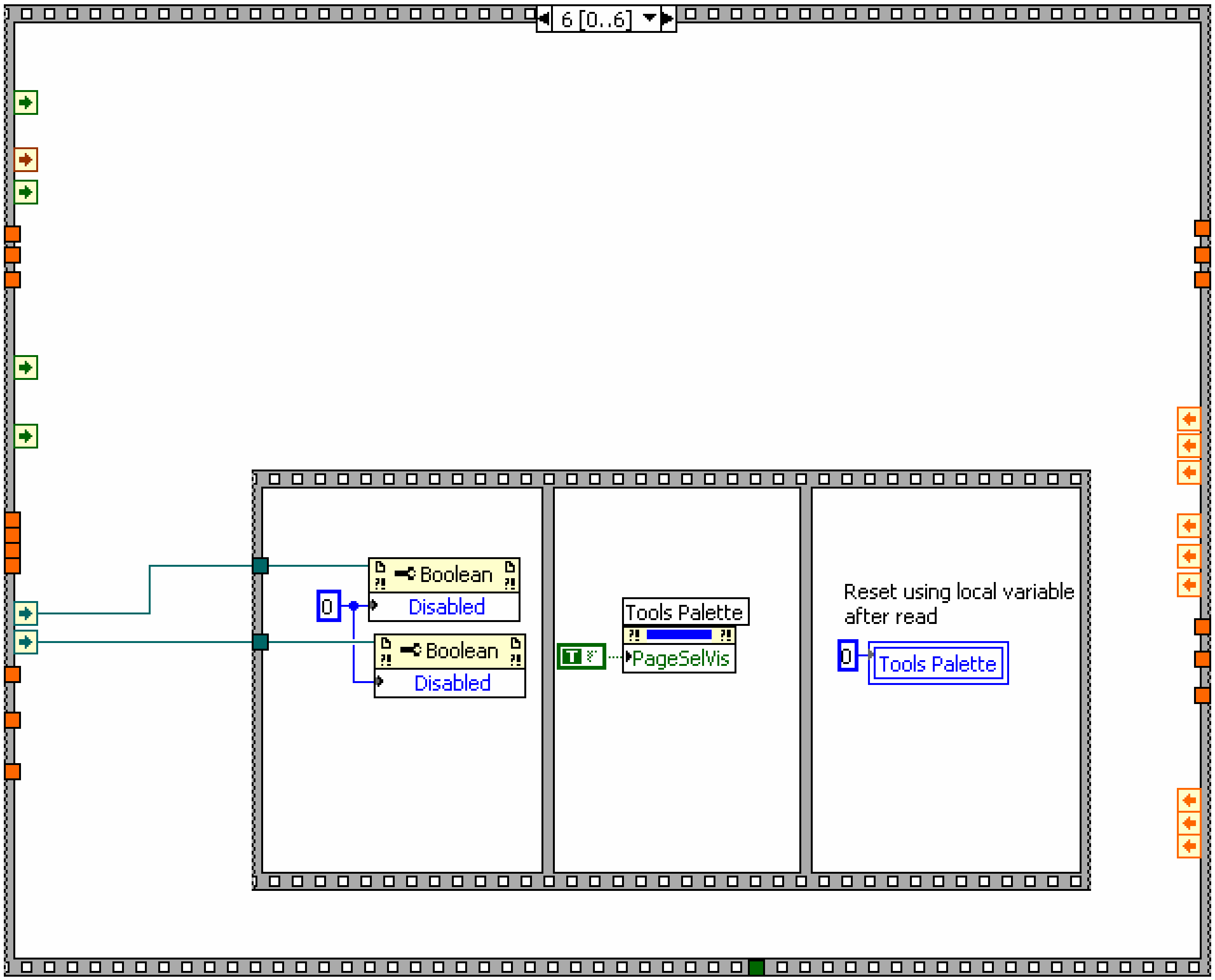

True
4 [0..6]
True
Wait for user to select
"Execute Motion" or "Cancel"
10
True
Path String
notify.wav
Play
Sound
Boolean
2
Disabled
Boolean
Disabled

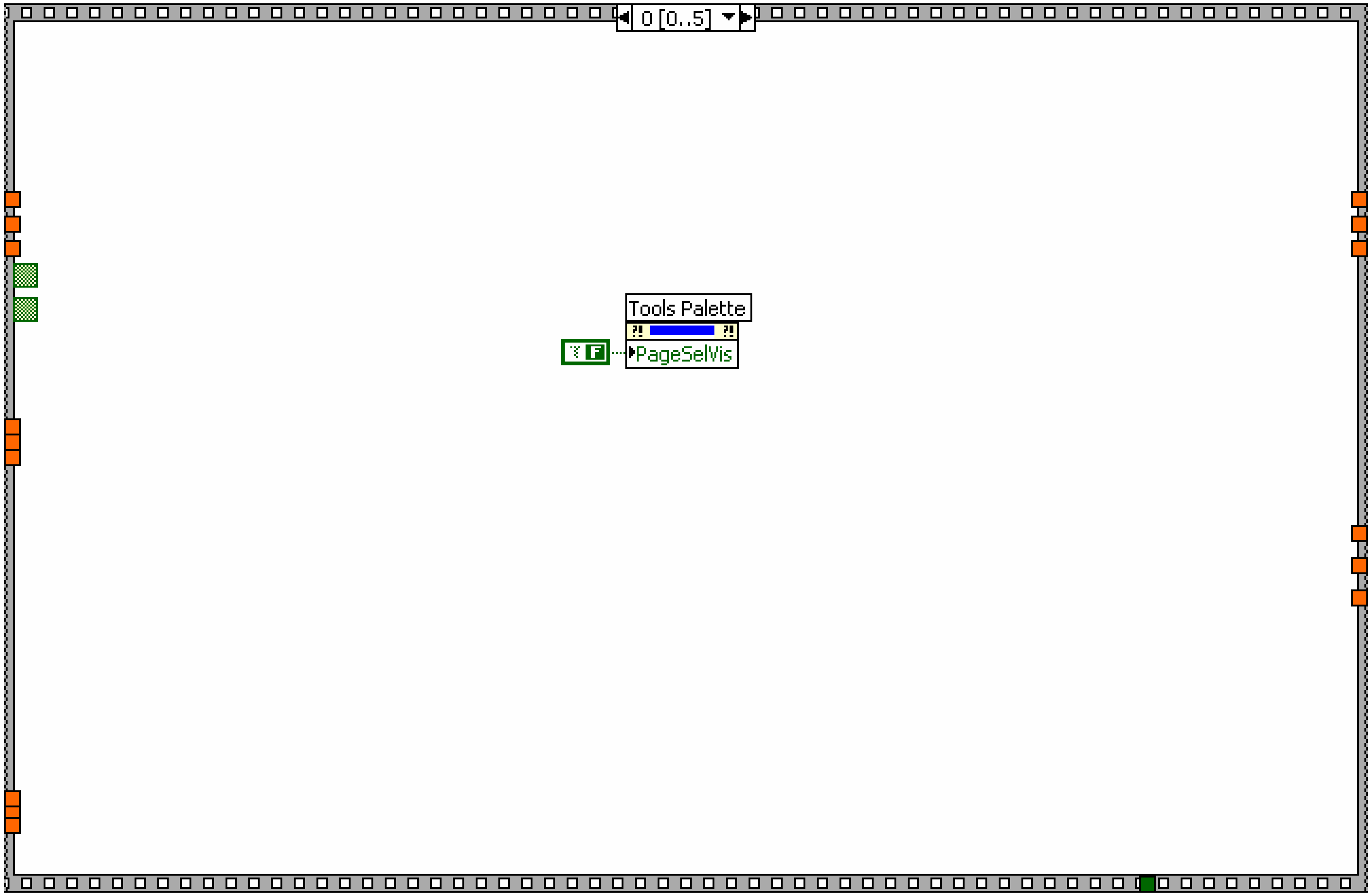

False
False
F

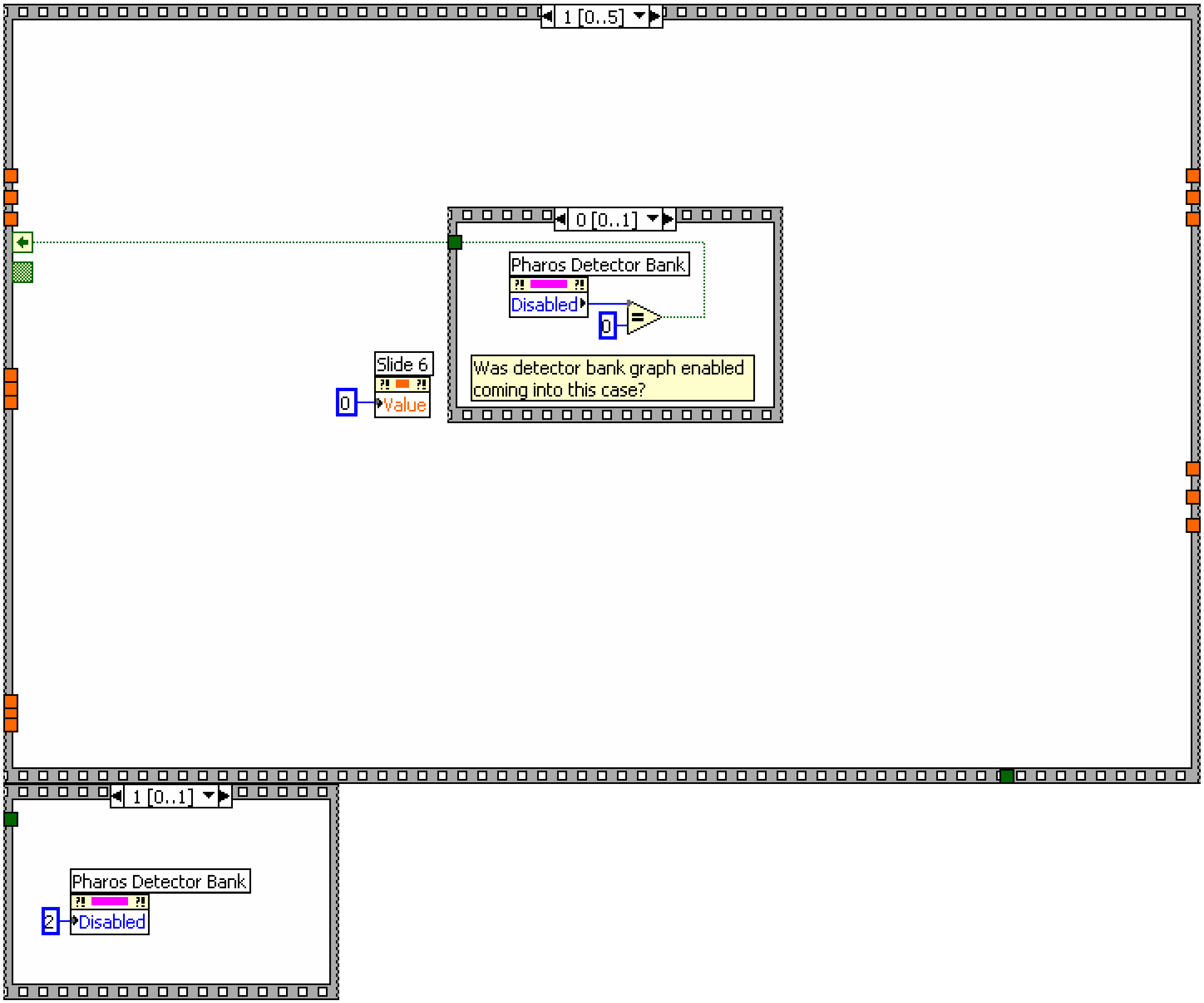

False
True
Pharos Detector Bank
2
Disabled
Detector bank graph was disabled coming into this function the user only did calculation but didnt move. Therefore, disable graph again.
True
Path String
Windows XP Recycle.wav
Play Sound
User has canceled execution of calculated motion: return current values for position and axes
F
False

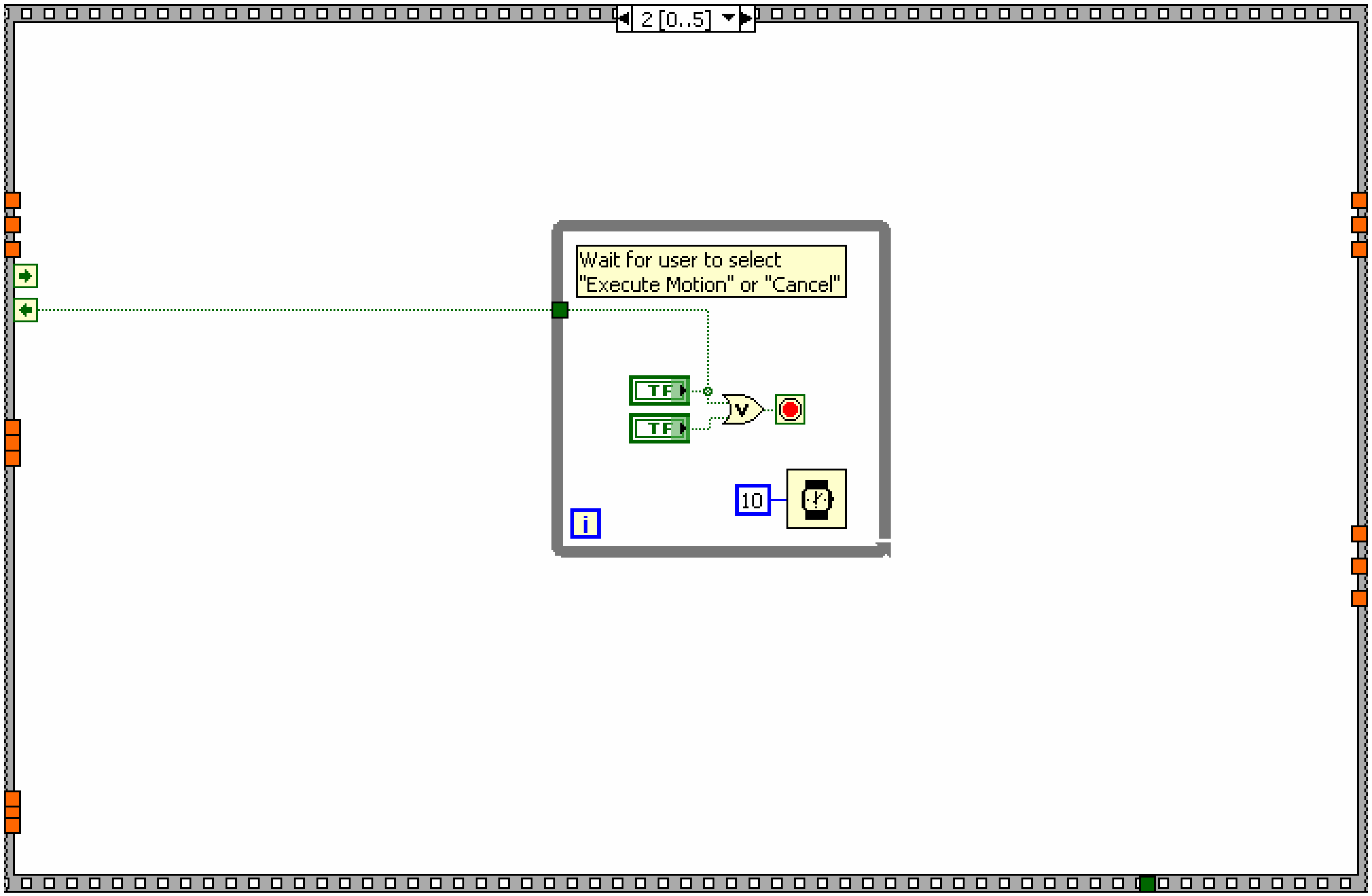

False
0 [0..2]
Calc Delay
1 [0..2]
Progress Bar
1
Slide 4
Scale.Maximum
Scale.Minimum
Minimum
Maximum
0
0
DBL
2
6K relative move

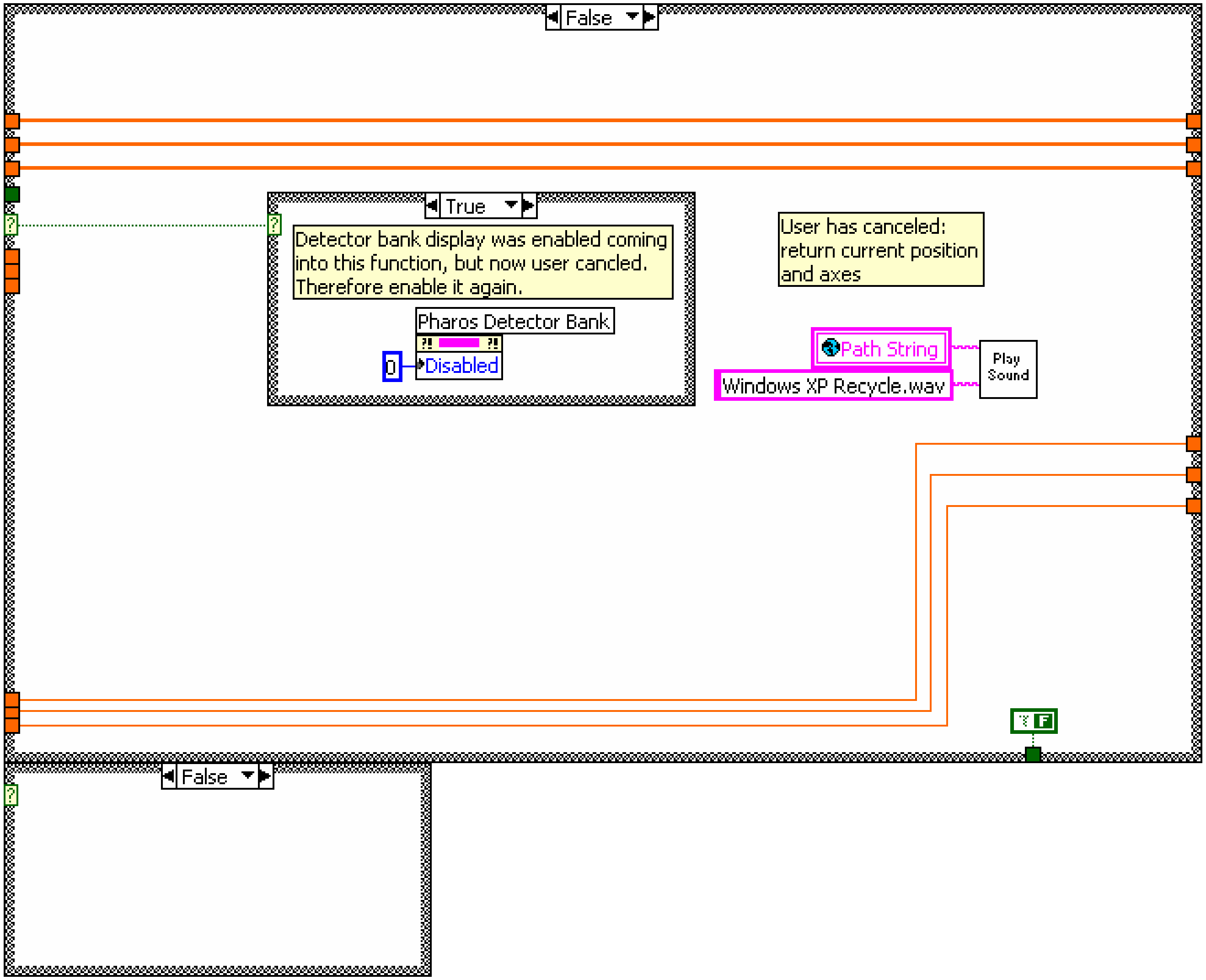

True
Comm
Error
Msg
?
T

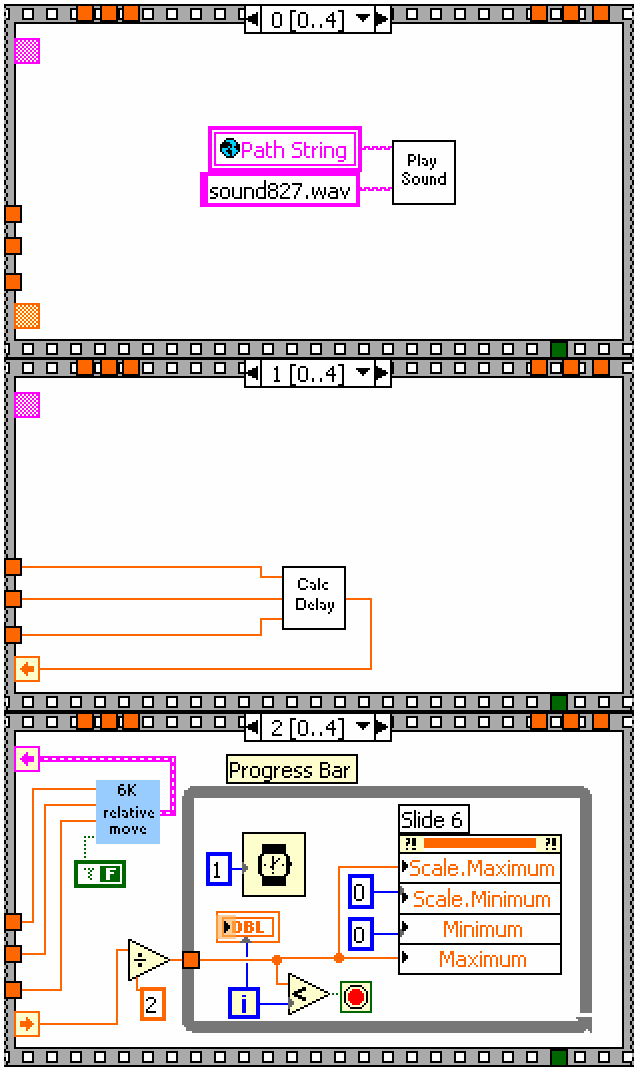

6 [0..6]
Boolean
Disabled
0
Boolean
Disabled
Tools Palette
PageSelVis
Reset using local variable after read
0
Tools Palette

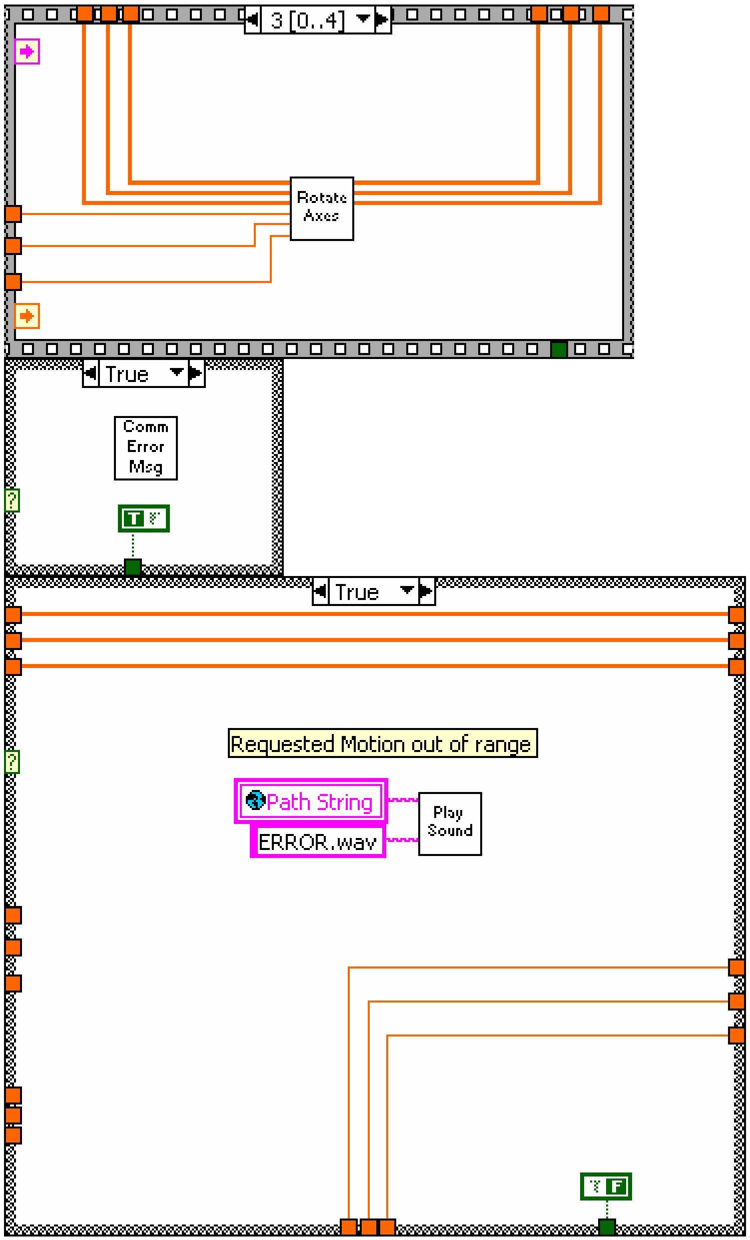

0 [0..5]
Tools Palette
PageSelVis
F

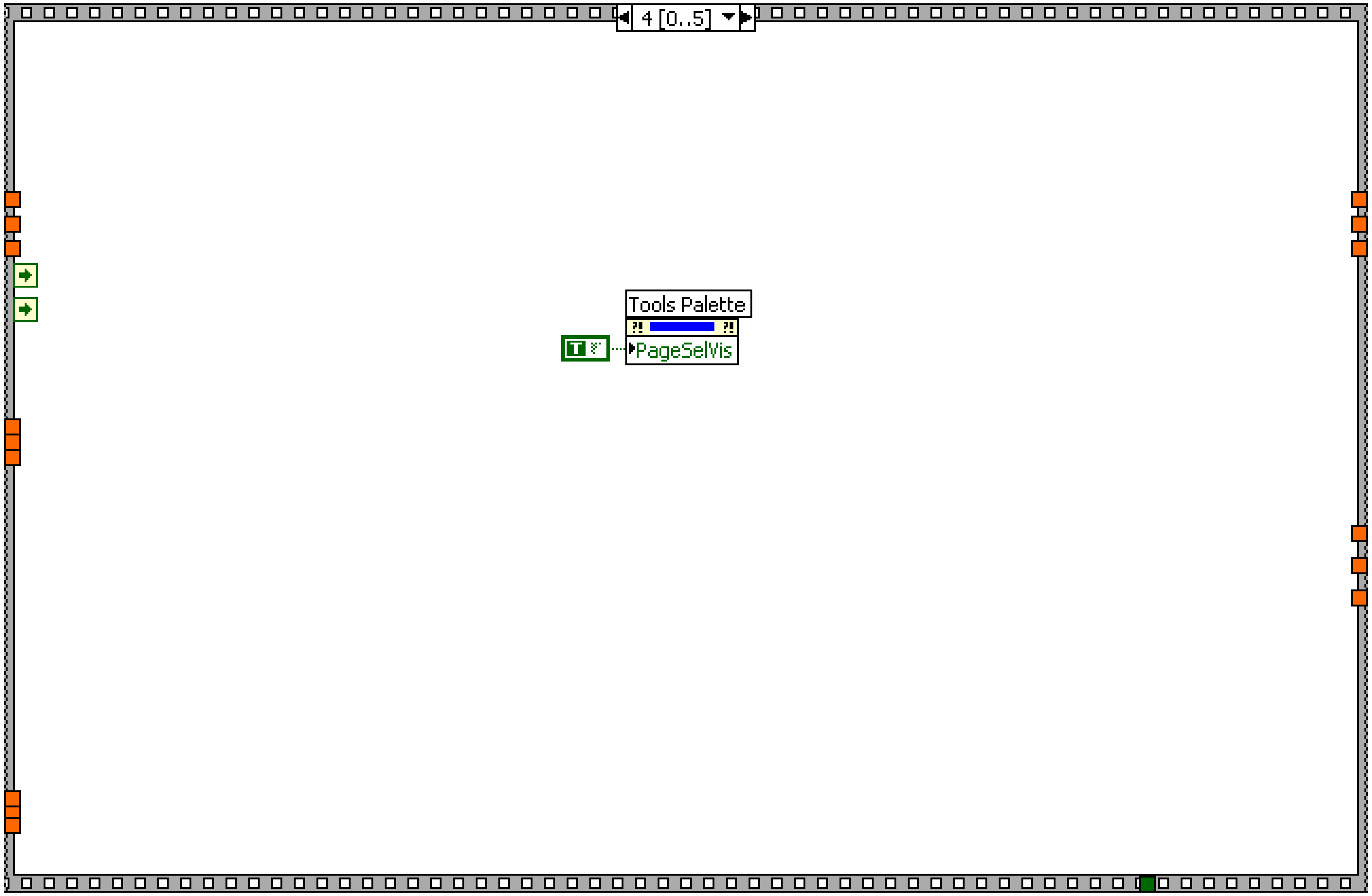

1 [0..5]
0 [0..1]
Pharos Detector Bank
Disabled
0
Slide 6
0
Value
Was detector bank graph enabled coming into this case?
1 [0..1]
Pharos Detector Bank
2
Disabled

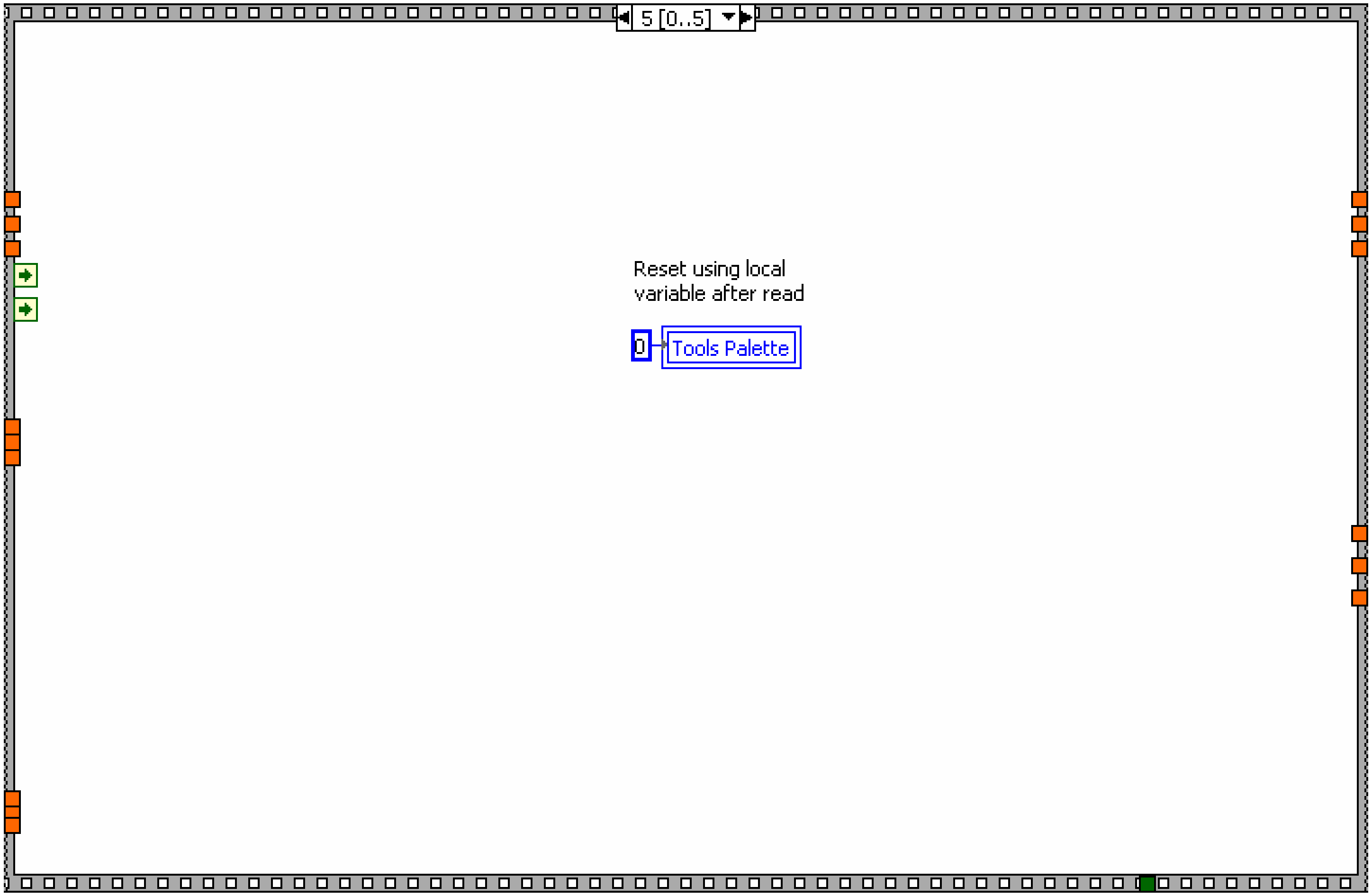

2 [0..5]
Wait for user to select
"Execute Motion" or "Cancel"
TF
TF
10

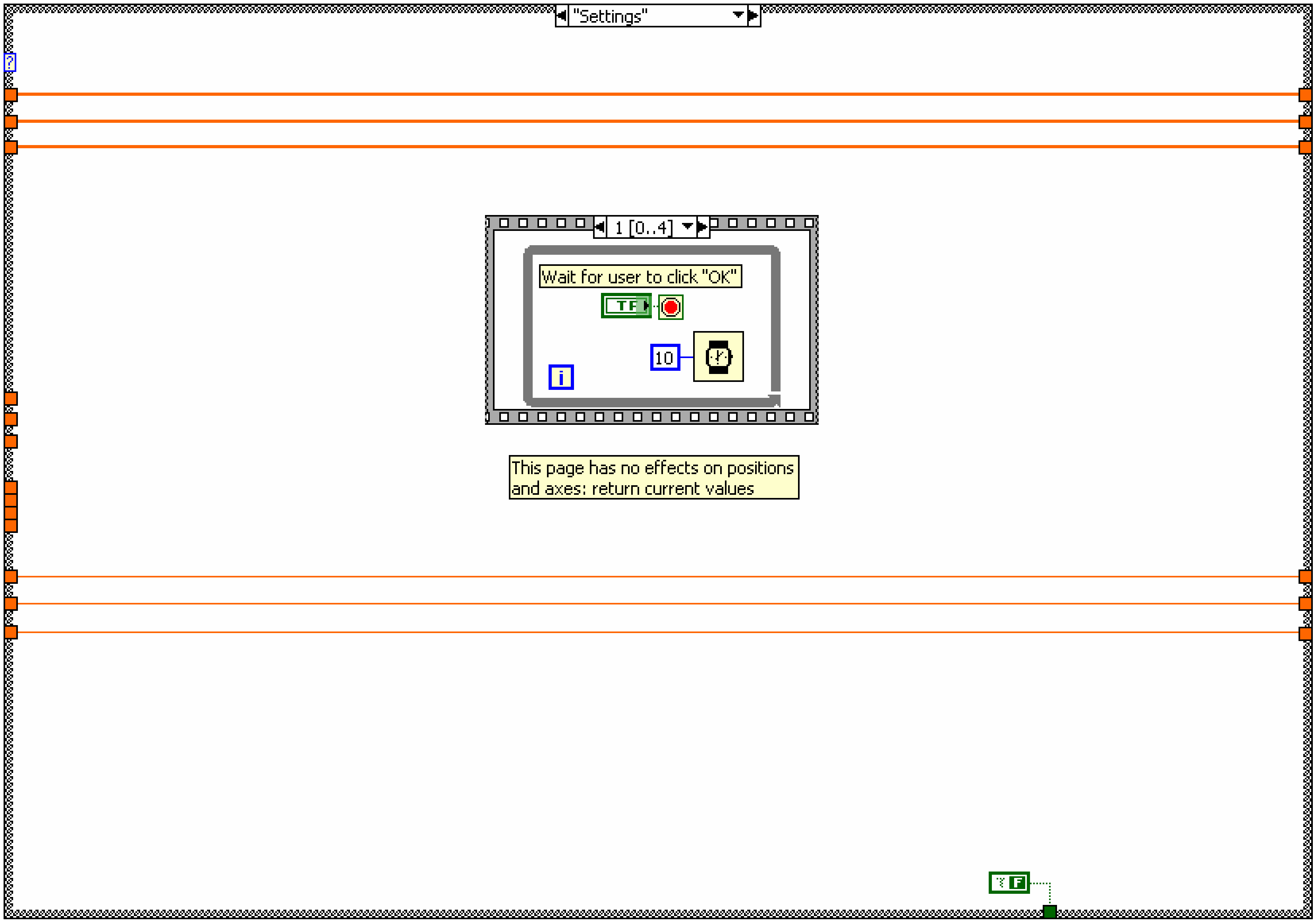

False
True
Detector bank display was enabled coming into this function, but now user cancled. Therefore enable it again.
Pharos Detector Bank
0
Disabled
User has canceled: return current position and axes
Path String
Windows XP Recycle.wav
Play Sound
False

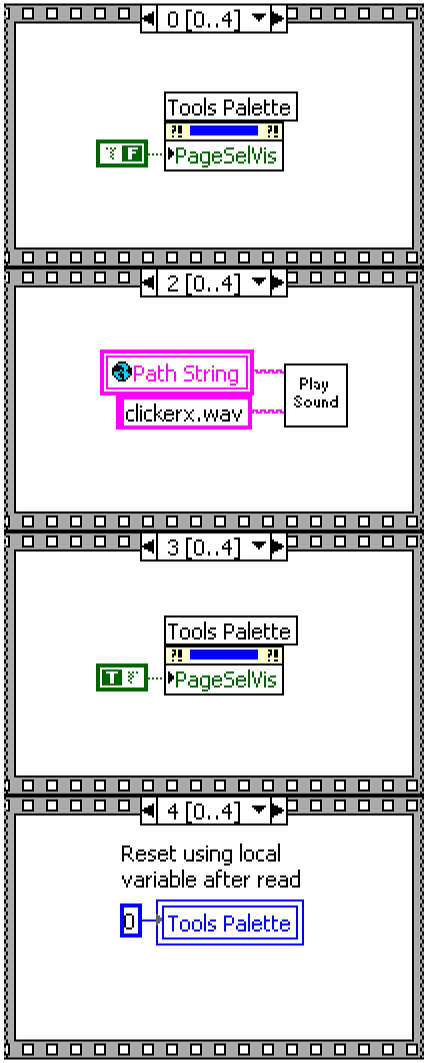

0 [0..4]
Path String
sound827.wav
Play Sound
1 [0..4]
Calc Delay
2 [0..4]
Progress Bar
6K relative move
Slide 6
Scale.Maximum
Scale.Minimum
Minimum
Maximum
DBL
1
0
0
2
i

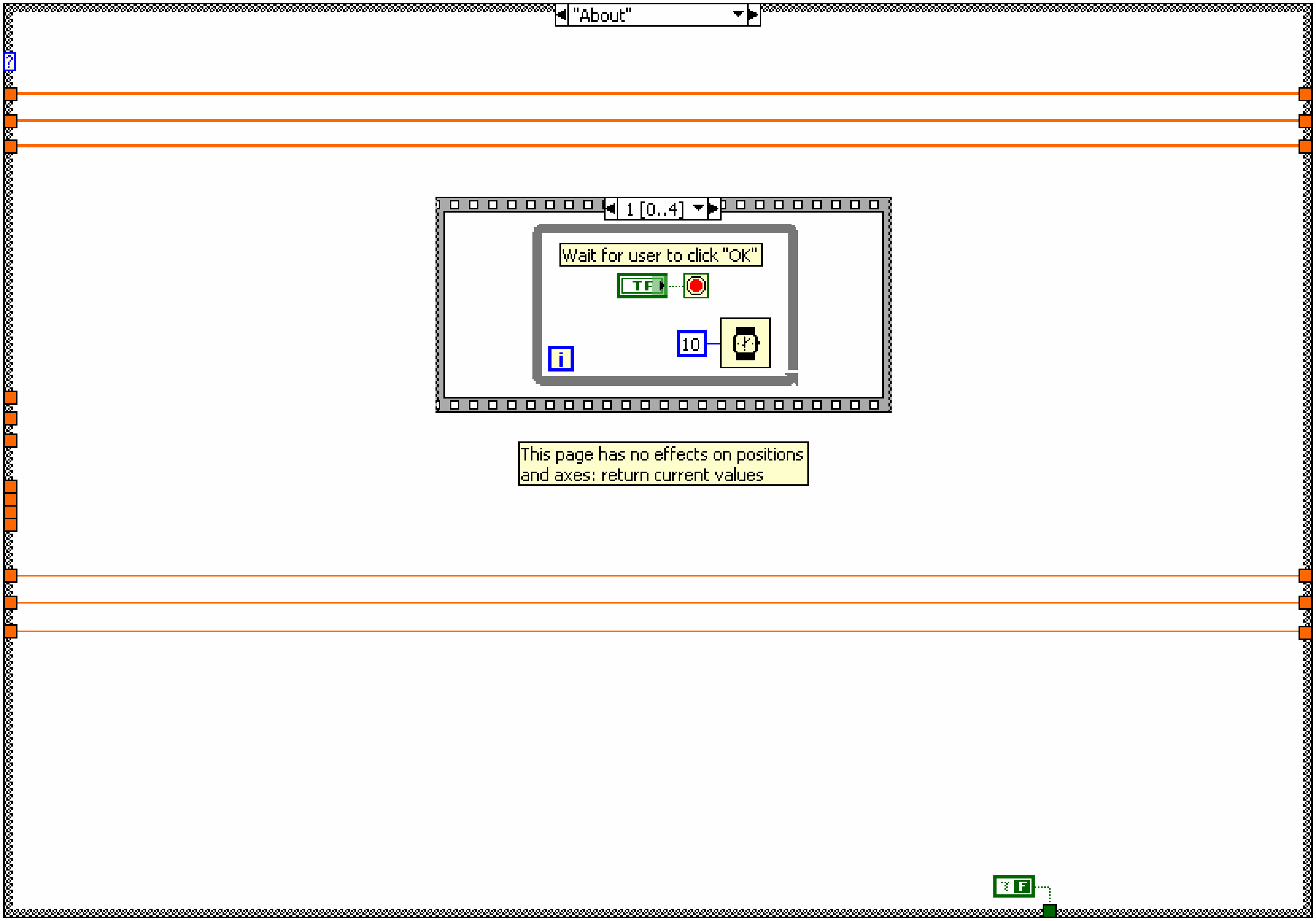

3 [0..4]
Rotate Axes
True
Comm Error Msg
True
Requested Motion out of range
Path String
ERROR.wav
Play Sound

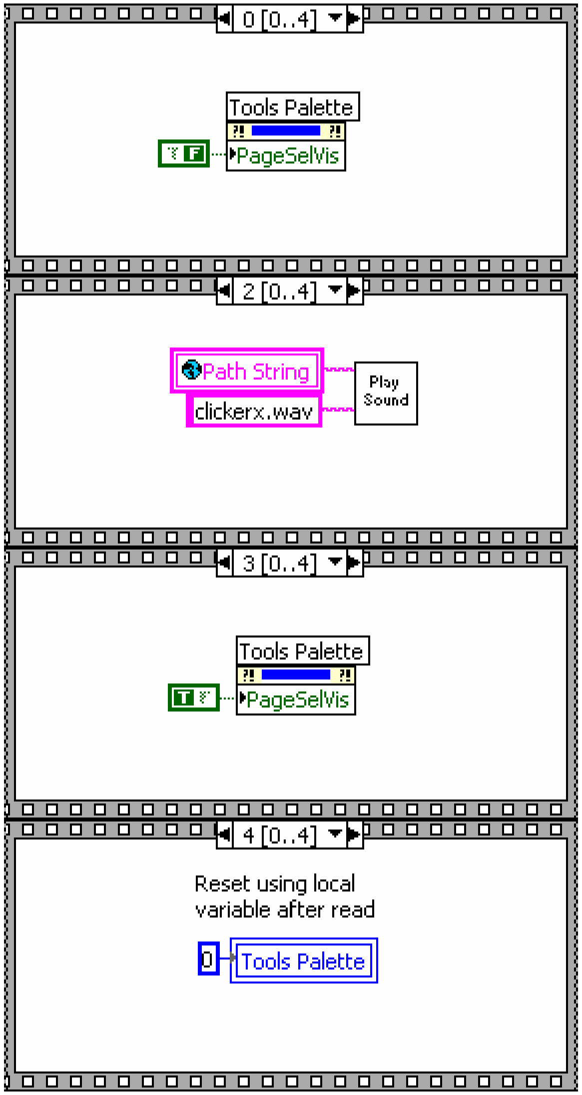

4 [0..5]
Tools Palette
PageSelVis

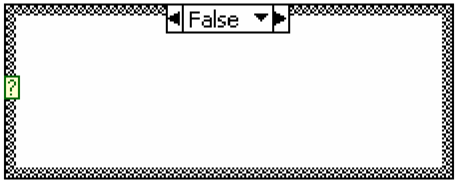

5 [0..5]
Reset using local
variable after read
0
Tools Palette


"Settings"
1 [0..4]
Wait for user to click "OK"
TF
10
i
This page has no effects on positions
and axes: return current values
F

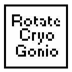

0 [0..4]
Tools Palette
PageSelVis
F
2 [0..4]
Path String
clickerx.wav
Play Sound
3 [0..4]
Tools Palette
PageSelVis
T
4 [0..4]
Reset using local variable after read
0
Tools Palette


"About"
1 [0..4]
Wait for user to click "OK"
TF
10
i
This page has no effects on positions and axes: return current values
TF

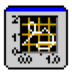

0 [0..4]
Tools Palette
PageSelVis
2 [0..4]
Path String
clickerx.wav
Play Sound
3 [0..4]
Tools Palette
PageSelVis
4 [0..4]
Reset using local variable after read
0
Tools Palette



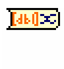
List of SubVIs and Express VIs with Configuration Information



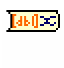
CallFastRotation.vi
D:\Copy of Goniometer VI\CallFastRotation.vi



Build XY Graph
Build XY Graph
formats the data displayed on an X-Y Graph.

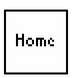


Convert to Dynamic Data
Convert to Dynamic Data

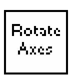


Convert to Dynamic Data2
Convert to Dynamic Data

6K Simple Move Basic_V3.vi
D:\Copy of Goniometer VI\6K Simple Move Basic_V3.vi



Home.vi
D:\Copy of Goniometer VI\Home.vi



RotateAxes.vi
D:\Copy of Goniometer VI\RotateAxes.vi

Display Message to User
Display Message to User
Displays a standard dialog box that contains an alert or a message for users.

This Express VI is configured as follows:

Message:

Display Message to User2
Display Message to User
Displays a standard dialog box that contains an alert or a message for users.

This Express VI is configured as follows:

Message:

3D_Graph.vi
D:\Copy of Goniometer VI\3D_Graph.vi

PlaySound.vi
D:\Copy of Goniometer VI\PlaySound.vi

Globals.vi
D:\Copy of Goniometer VI\Globals.vi

CalculateDelay.vi
D:\Copy of Goniometer VI\CalculateDelay.vi

SumAndRangeCheck.vi
D:\Copy of Goniometer VI\SumAndRangeCheck.vi

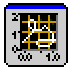
CurrentDirectoryString.vi
D:\Copy of Goniometer VI\CurrentDirectoryString.vi

LimitAll.vi
D:\Copy of Goniometer VI\LimitAll.vi

Display Message to User3
Display Message to User
Displays a standard dialog box that contains an alert or a message for users.

This Express VI is configured as follows:

Message:



Build XY Graph2
Build XY Graph
formats the data displayed on an X-Y Graph.

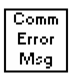
Convert to Dynamic Data3
Convert to Dynamic Data

Convert to Dynamic Data4
Convert to Dynamic Data

Display Message to User4
Display Message to User
Displays a standard dialog box that contains an alert or a message for users.

This Express VI is configured as follows:

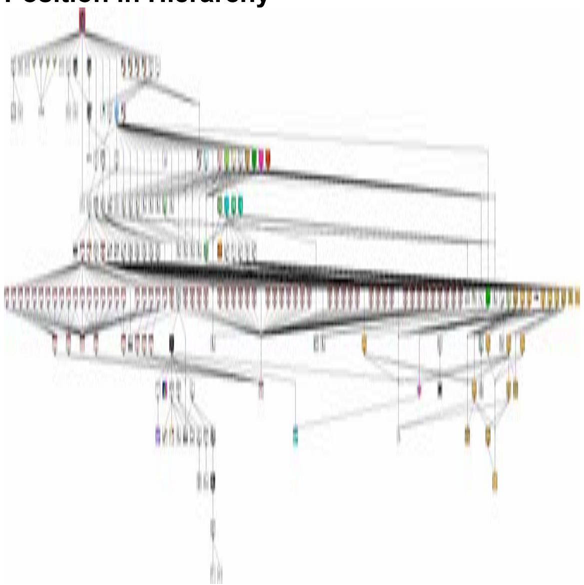
Message:



CommErrorMessage.vi
D:\Copy of Goniometer VI\CommErrorMessage.vi

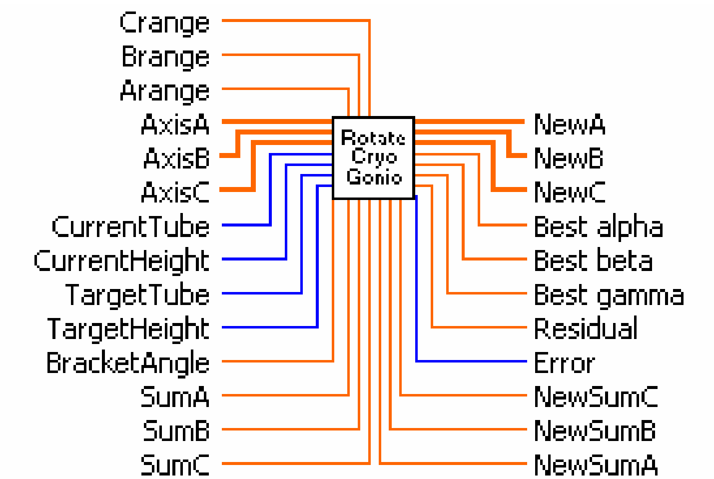
VI Revision History
"GoniometerMotionControl.vi History"
Current Revision: 0
rev. 0 Wed, Nov 10, 2004 9:46:00 AM robert
Version 1.0 Release, Markus Hehlen, LANSCE-12, 11/10/04

Position in Hierarchy

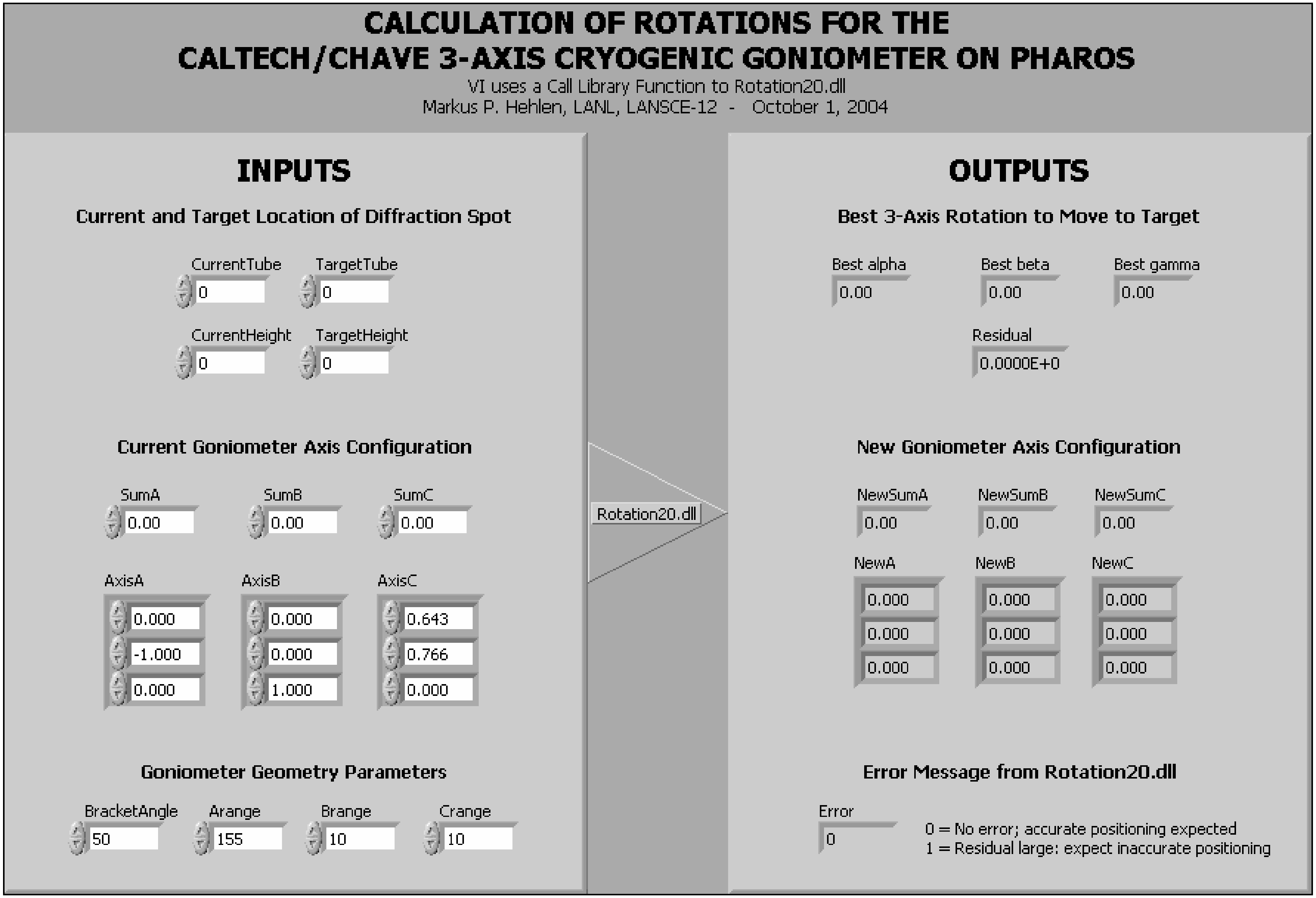


CallFastRotation.vi

Connector Pane



Front Panel


CALCULATION OF ROTATIONS FOR THE
CALTECH/CHAVE 3-AXIS CRYOGENIC GONIOMETER ON PHAROS
VI uses a Call Library Function to Rotation20.dll
Markus P. Hehlen, LANL, LANSCE-12 - October 1, 2004
INPUTS
Current and Target Location of Diffraction Spot
CurrentTube
0
TargetTube
0
CurrentHeight
0
TargetHeight
0
Current Goniometer Axis Configuration
SumA
0.00
SumB
0.00
SumC
0.00
AxisA
0.000
-1.000
0.000
AxisB
0.000
0.000
1.000
AxisC
0.643
0.766
0.000
Goniometer Geometry Parameters
BracketAngle
50
Arange
155
Brange
10
Crange
10
Rotation20.dll
OUTPUTS
Best 3-Axis Rotation to Move to Target
Best alpha
0.00
Best beta
0.00
Best gamma
0.00
Residual
0.0000E+0
New Goniometer Axis Configuration
NewSumA
0.00
NewSumB
0.00
NewSumC
0.00
NewA
0.000
0.000
0.000
NewB
0.000
0.000
0.000
NewC
0.000
0.000
0.000
Error Message from Rotation20.dll
Error
0
0 = No error; accurate positioning expected
1 = Residual large: expect inaccurate positioning

Controls and Indicators

[DBL] **AxisA**

  DBL **Numeric**

[DBL] **AxisB**

  DBL **Numeric**

[DBL] **AxisC**

  DBL **Numeric**

DBL **BracketAngle**

DBL **Arange**

DBL **Brange**

DBL **Crange**

DBL **SumA**

DBL **SumB**

DBL **SumC**

I32 **CurrentTube**

I32 **CurrentHeight**

I32 **TargetTube**

I32 **TargetHeight**

DBL **Best alpha**

DBL **Best beta**

DBL **Best gamma**

DBL **Residual**

[DBL] **NewA**

- DBL **Numeric**

[DBL] **NewB**

- DBL **Numeric**

[DBL] **NewC**

- DBL **Numeric**

I32 **Error**

DBL **NewAx**

DBL **NewAy**

DBL **NewAz**

DBL **NewBx**

DBL **NewBy**

DBL **NewBz**

DBL **NewCx**

DBL **NewCy**

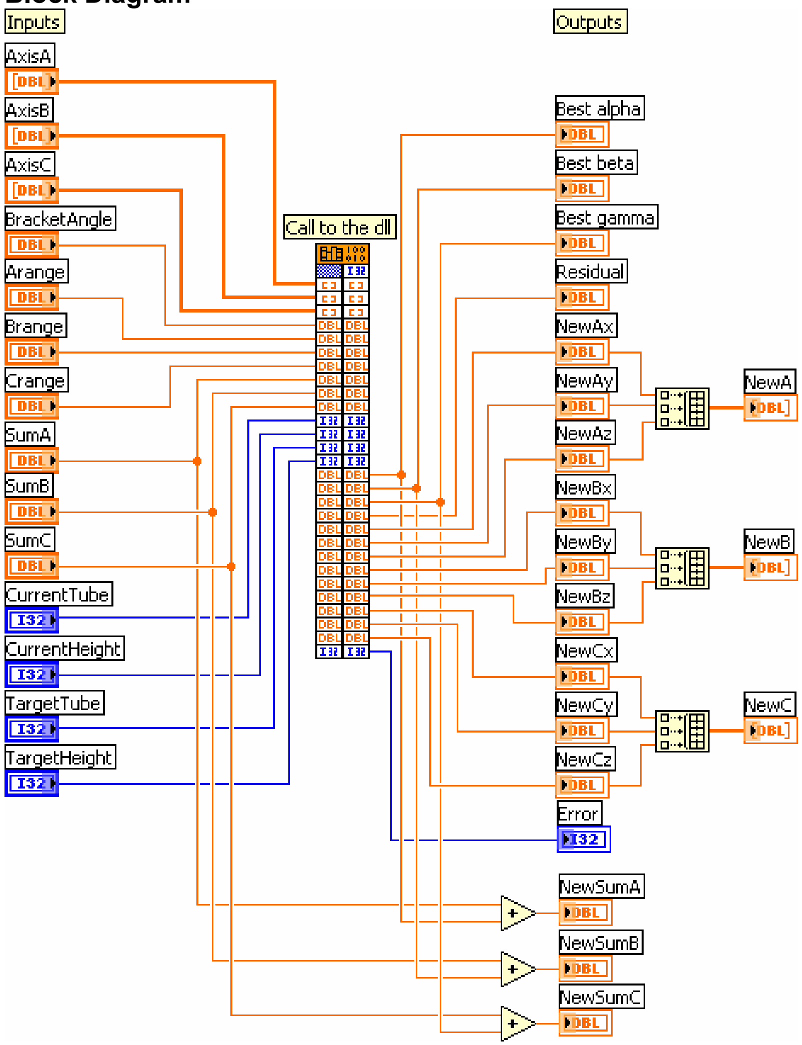
DBL **NewCz**

DBL **NewSumA**

DBL **NewSumB**

DBL **NewSumC**

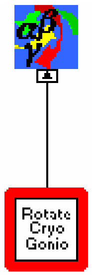
Block Diagram



List of SubVIs and Express VIs with Configuration Information

VI Revision History
"CallFastRotation.vi History"
Current Revision: 40

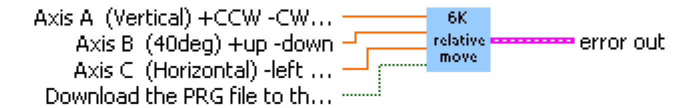
Position in Hierarchy



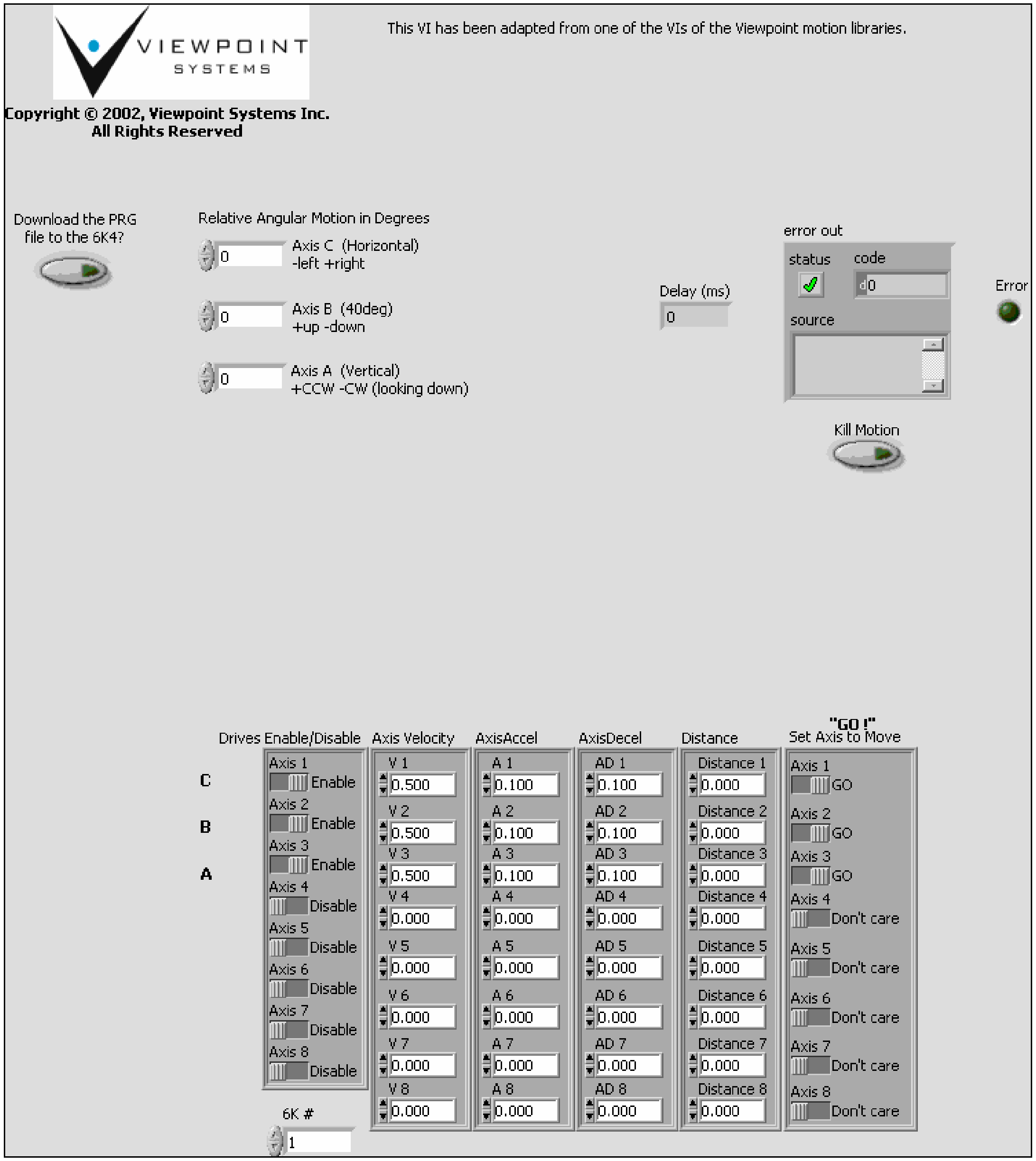
6K Simple Move Basic_V3.vi

Author: Alan Sharrow

Connector Pane



Front Panel


VIEWPOINT SYSTEMS
Copyright © 2002, Viewpoint Systems Inc.
All Rights Reserved
This VI has been adapted from one of the VIs of the Viewpoint motion libraries.
Download the PRG file to the 6K4?
Relative Angular Motion in Degrees
Axis C (Horizontal) -left +right
Axis B (40deg) +up -down
Axis A (Vertical) +CCW -CW (looking down)
Delay (ms)
error out
status
code
source
Error
Kill Motion
Drives Enable/Disable
Axis Velocity
AxisAccel
AxisDecel
Distance
"GO !" Set Axis to Move
C
B
A
Axis 1 Enable
Axis 2 Enable
Axis 3 Enable
Axis 4 Disable
Axis 5 Disable
Axis 6 Disable
Axis 7 Disable
Axis 8 Disable
V 1 0.500
V 2 0.500
V 3 0.500
V 4 0.000
V 5 0.000
V 6 0.000
V 7 0.000
V 8 0.000
A 1 0.100
A 2 0.100
A 3 0.100
A 4 0.000
A 5 0.000
A 6 0.000
A 7 0.000
A 8 0.000
AD 1 0.100
AD 2 0.100
AD 3 0.100
AD 4 0.000
AD 5 0.000
AD 6 0.000
AD 7 0.000
AD 8 0.000
Distance 1 0.000
Distance 2 0.000
Distance 3 0.000
Distance 4 0.000
Distance 5 0.000
Distance 6 0.000
Distance 7 0.000
Distance 8 0.000
Axis 1 GO
Axis 2 GO
Axis 3 GO
Axis 4 Don't care
Axis 5 Don't care
Axis 6 Don't care
Axis 7 Don't care
Axis 8 Don't care
6K #
1

Controls and Indicators

6K # Number of the 6K that you want to set velocity on.

Set Velocities Set Velocities cluster. You can specify which axes that you want to set velocity on and which axes you want to ignore and the velocity parameters for the axes that you selected.
NOTE: Velcoity units can be in raw counts/sec, revs/sec, inches/sec or millimeters/second depending upon how you configured your 6K system (SCLV) in the 6L .prg program file.

- **Set Axis**
  - **Axis 1**
  - **Axis 2**
  - **Axis 3**
  - **Axis 4**
  - **Axis 5**
  - **Axis 6**
  - **Axis 7**
  - **Axis 8**
- **Axis Velocity**
  - **V 1**
  - **V 2**
  - **V 3**
  - **V 4**
  - **V 5**

DBL **V 6**

DBL **V 7**

DBL **V 8**

Set Accel Set Accel Cluster. Set the Axis that you want to write the Accel parameters to. Set the Accel parameter. NOTE: This is based upon how you defined Motion Scaling in your 6K. Units of Revs/Sec2, Inches/Sec2, Millimeters/Sec2 or raw counts/Sec2.

Set Axis

TF **Axis 1**

TF **Axis 2**

TF **Axis 3**

TF **Axis 4**

TF **Axis 5**

TF **Axis 6**

TF **Axis 7**

TF **Axis 8**

AxisAccel

DBL **A 1**

DBL **A 2**

DBL **A 3**

DBL **A 4**

DBL **A 5**

A 6

A 7

A 8

Set Decels Set Decel Cluster. Set the Axis that you want to write the Decel parameters to. Set the Decel parameter. NOTE: This is based upon how you defined Motion Scaling in your 6K. Units of Revs/Sec2, Inches/Sec2, Millimeters/Sec2 or raw counts/Sec2.

Set Axis

Axis 1

Axis 2

Axis 3

Axis 4

Axis 5

Axis 6

Axis 7

Axis 8

AxisDecel

AD 1

AD 2

AD 3

AD 4

DBL **AD 5**

DBL **AD 6**

DBL **AD 7**

DBL **AD 8**

Set Distance & Direction Set Distance Cluster. You can specify the axes that you want to set distance and direction parameters and the axes that you want to ignore. You can specify CW or CCW direction on each axis and the Distance parameter for each axis. NOTE: That the distance units is determined by how you chose to setup distance scaling in the 6K (SCLD). Distance units may be in raw counts, revs, inches or millimeters depending upon how you specified scaling.

Set Axis

TF **Axis 1**

TF **Axis 2**

TF **Axis 3**

TF **Axis 4**

TF **Axis 5**

TF **Axis 6**

TF **Axis 7**

TF **Axis 8**

Set Direction

TF **Axis 1**

TF **Axis 2**

        - **Axis 3**
        - **Axis 4**
        - **Axis 5**
        - **Axis 6**
        - **Axis 7**
        - **Axis 8**
    - **Distance**
        - **Distance 1**
        - **Distance 2**
        - **Distance 3**
        - **Distance 4**
        - **Distance 5**
        - **Distance 6**
        - **Distance 7**
        - **Distance 8**
- **Set Axis to Move**
    - **Axis 1**
    - **Axis 2**
    - **Axis 3**
    - **Axis 4**

- **Axis 5**
- **Axis 6**
- **Axis 7**
- **Axis 8**

Drives Enable/Disable Axis Select Cluster to enable or disable specific drives for 1 thru. 8 axes.

- **Axis 1**
- **Axis 2**
- **Axis 3**
- **Axis 4**
- **Axis 5**
- **Axis 6**
- **Axis 7**
- **Axis 8**

Axis Velocity

- **V 1**
- **V 2**
- **V 3**
- **V 4**
- **V 5**

- DBL **V 6**
- DBL **V 7**
- DBL **V 8**

Distance

- DBL **Distance 1**
- DBL **Distance 2**
- DBL **Distance 3**
- DBL **Distance 4**
- DBL **Distance 5**
- DBL **Distance 6**
- DBL **Distance 7**
- DBL **Distance 8**

AxisAccel

- DBL **A 1**
- DBL **A 2**
- DBL **A 3**
- DBL **A 4**
- DBL **A 5**
- DBL **A 6**

A 7

A 8

AxisDecel

AD 1

AD 2

AD 3

AD 4

AD 5

AD 6

AD 7

AD 8

Axis C (Horizontal) -left +right

Axis B (40deg) +up -down

Axis A (Vertical) +CCW -CW (looking down)

Kill Motion

Download the PRG file to the 6K4?

error out The **error out** cluster passes error or warning information out of a VI to be used by other VIs.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

status The **status** boolean is either TRUE (X) for an error, or FALSE (checkmark) for no error or a warning.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

code The **code** input identifies the error or warning.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

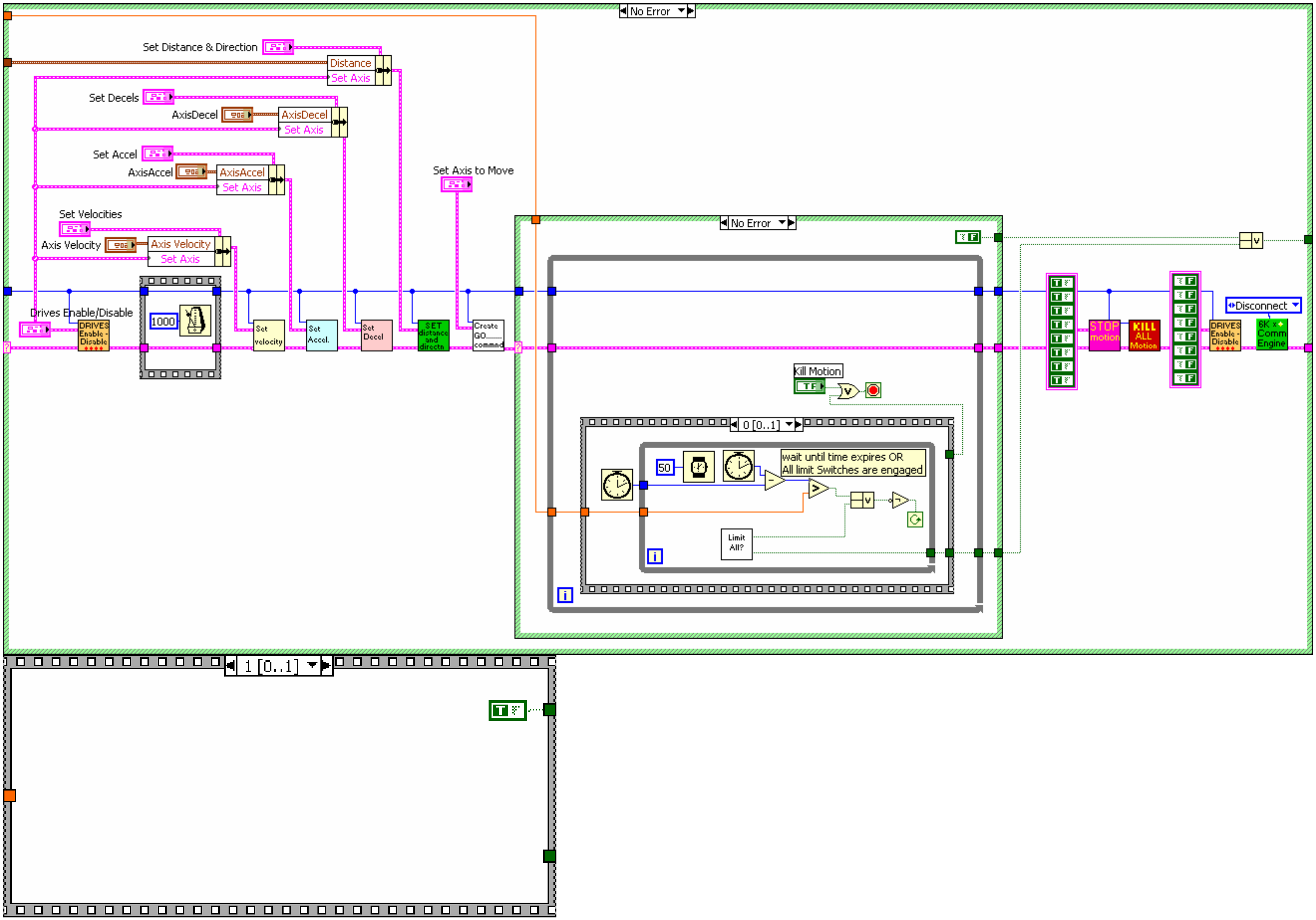
source The **source** string describes the origin of the error or warning.

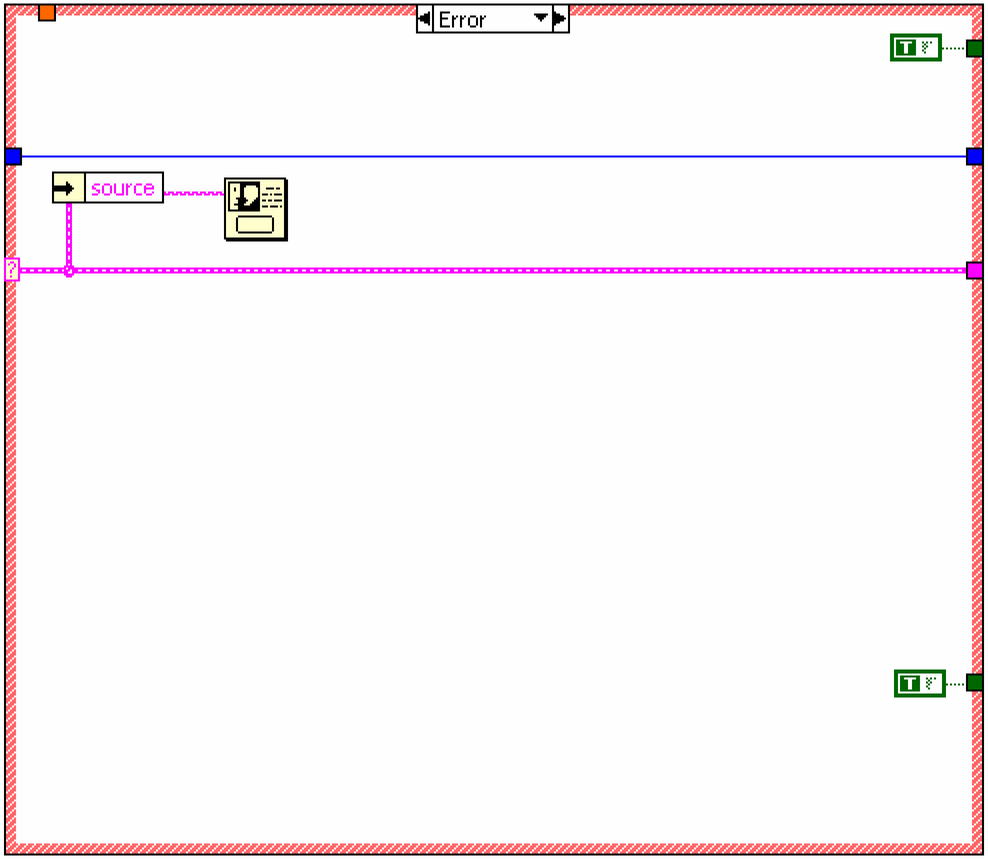
The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

Error

Delay (ms)

Block Diagram

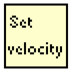

No Error
Set Distance & Direction
Distance
Set Axis
Set Decels
AxisDecel
AxisDecel
Set Axis
Set Accel
AxisAccel
AxisAccel
Set Axis
Set Axis to Move
Set Velocities
Axis Velocity
Axis Velocity
Set Axis
Drives Enable/Disable
DRIVES Enable - Disable
1000
Set velocity
Set Accel.
Set Decel
SET distance and directn
Create GO...... commnd
No Error
Kill Motion
0 [0..1]
50
wait until time expires OR
All limit Switches are engaged
Limit All?
STOP motion
KILL ALL Motion
DRIVES Enable - Disable
Disconnect
6K Comm Engine
1 [0..1]



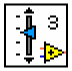
List of SubVIs and Express VIs with Configuration Information

6Kx Comm Launcher.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\6Kx Comm Launcher.vi

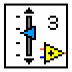
minimize path.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\minimize path.vi



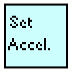
Set Velocity.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Velocity.vi



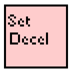
8 Axis Velocity Floats.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\8 Axis Velocity Floats.ctl



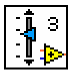
Set Velocity.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Velocity.ctl



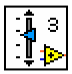
Set Accel.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Accel.vi



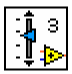
Set Decel.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Decel.vi



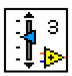
8 Axis Accel Floats.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\8 Axis Accel Floats.ctl



Set Acceleration.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Acceleration.ctl

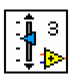


8 Axis Decel Floats.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\8 Axis Decel Floats.ctl

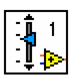


Set Deceleration.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Deceleration.ctl

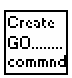
Set Distance+Direction.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Distance+Direction.vi



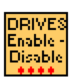
Set Distance.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\Set Distance.ctl



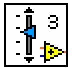
Pick Axes.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\Pick Axes.ctl



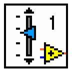
Create GO command.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\Create GO command.vi



Enable-Disable Drives.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\Enable-Disable Drives.vi



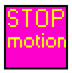
Enable-Disable Drives.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\Enable-Disable Drives.ctl



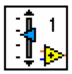
6Kx Comm Engine Actions.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\6Kx Comm Engine Actions.ctl

6Kx Comm Engine.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\6Kx Comm Engine.vi

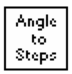
KILL Motion.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\KILL Motion.vi



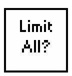
STOP Motion.vi
D:\Copy of Goniometer VI\Viewpoint Motion Library\STOP Motion.vi



8 Axis Booleans.ctl
D:\Copy of Goniometer VI\Viewpoint Motion Library\8 Axis Booleans.ctl



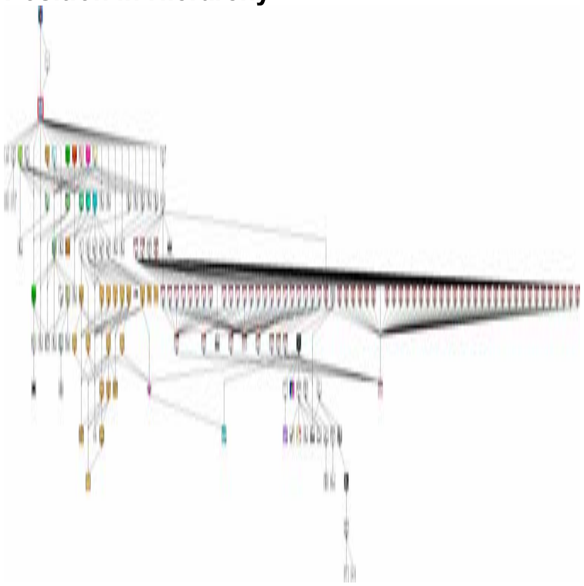
AngleToSteps.vi
D:\Copy of Goniometer VI\AngleToSteps.vi



LimitAll.vi
D:\Copy of Goniometer VI\LimitAll.vi

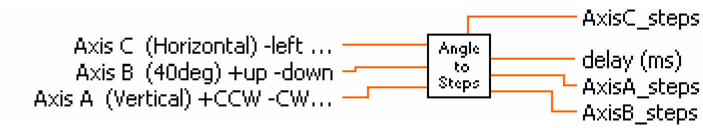
VI Revision History
"6K Simple Move Basic_V3.vi History"
Current Revision: 82

Position in Hierarchy

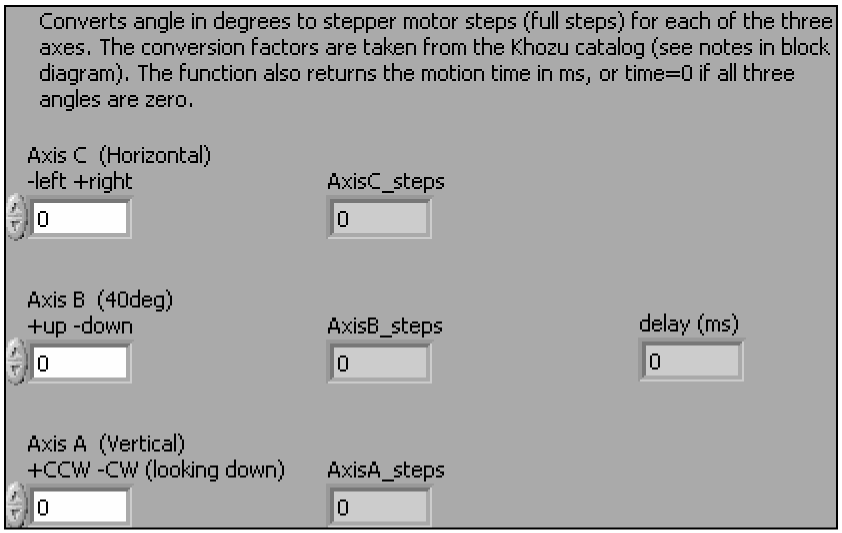


AngleToSteps.vi

Connector Pane



Front Panel



Controls and Indicators

DBL **Axis A (Vertical) +CCW -CW (looking down)**

DBL **Axis B (40deg) +up -down**

DBL **Axis C (Horizontal) -left +right**

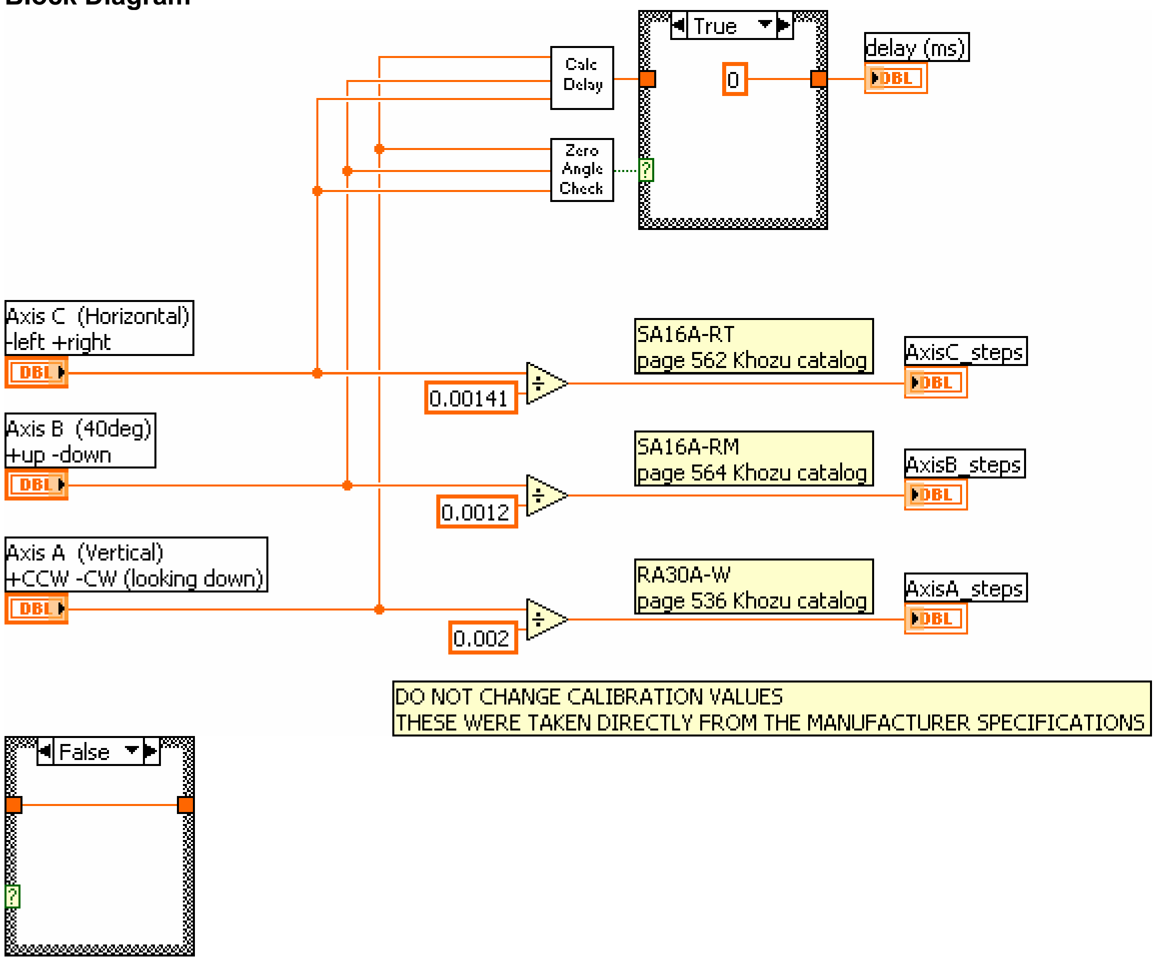
DBL **delay (ms)**

DBL **AxisC_steps**

DBL **AxisB_steps**

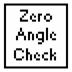
DBL **AxisA_steps**

Block Diagram



List of SubVIs and Express VIs with Configuration Information

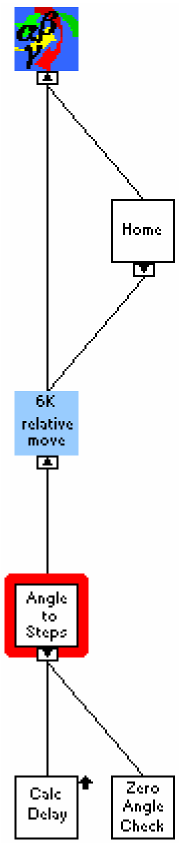
CalculateDelay.vi
D:\Copy of Goniometer VI\CalculateDelay.vi



ZeroAngleCheck.vi
D:\Copy of Goniometer VI\ZeroAngleCheck.vi

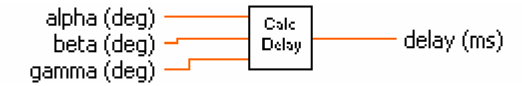
VI Revision History
"AngleToSteps.vi History"
Current Revision: 4

Position in Hierarchy

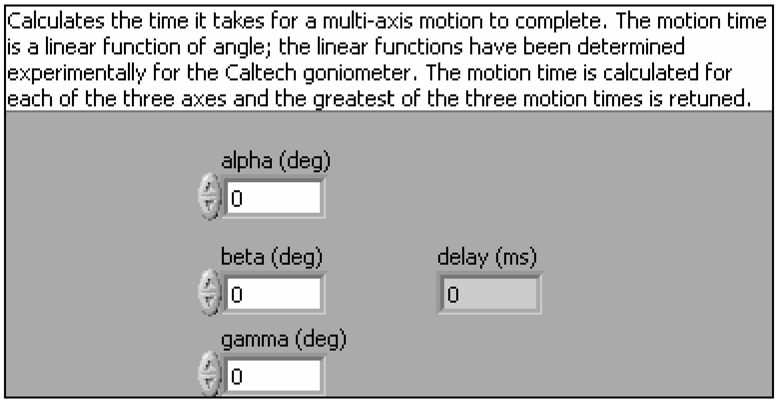


CalculateDelay.vi

Connector Pane



Front Panel



Controls and Indicators

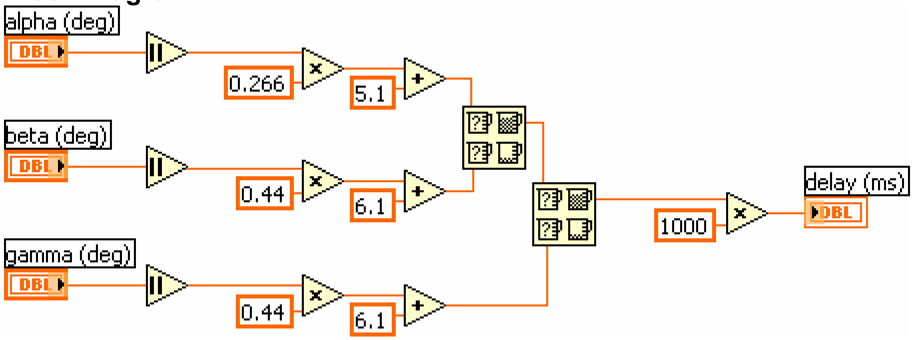
DBL **gamma (deg)**

DBL **beta (deg)**

DBL **alpha (deg)**

DBL **delay (ms)**

Block Diagram



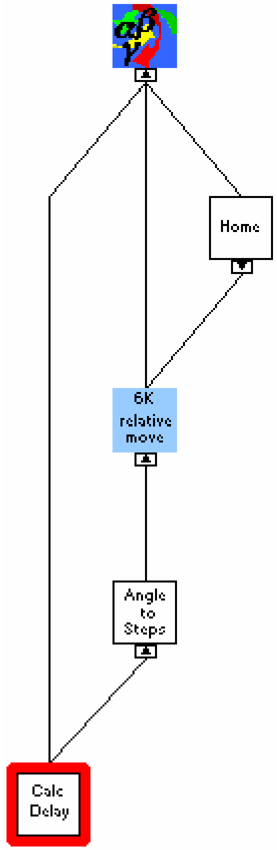
List of SubVIs and Express VIs with Configuration Information

VI Revision History

"CalculateDelay.vi History"

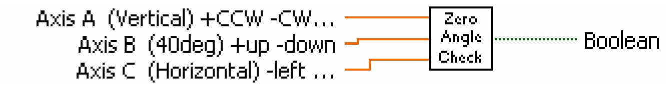
Current Revision: 5

Position in Hierarchy

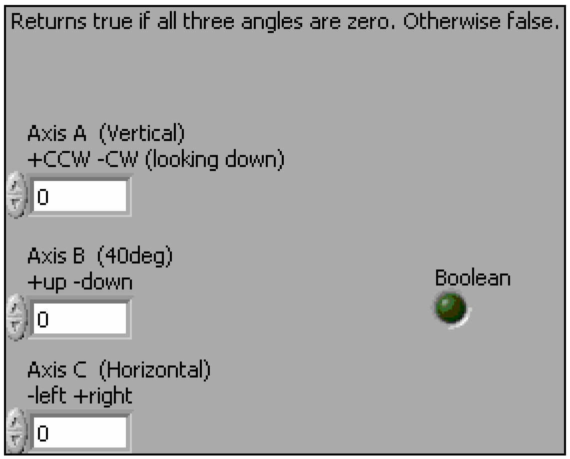


ZeroAngleCheck.vi

Connector Pane



Front Panel



Controls and Indicators

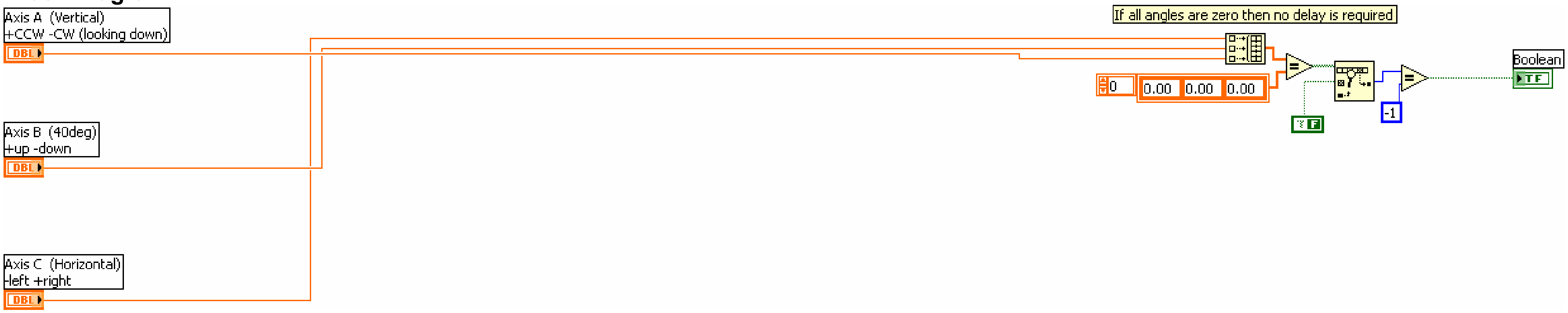
DBL **Axis C (Horizontal) -left +right**

DBL **Axis B (40deg) +up -down**

DBL **Axis A (Vertical) +CCW -CW (looking down)**

TF **Boolean**

Block Diagram

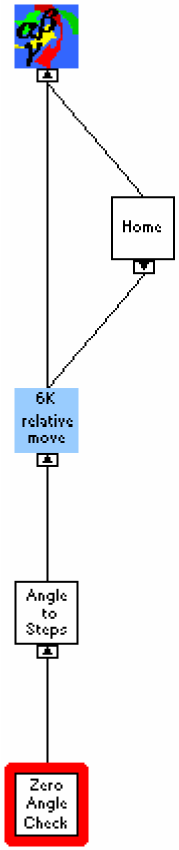


List of SubVIs and Express VIs with Configuration Information

VI Revision History

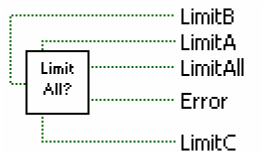
"ZeroAngleCheck.vi History"
Current Revision: 3

Position in Hierarchy

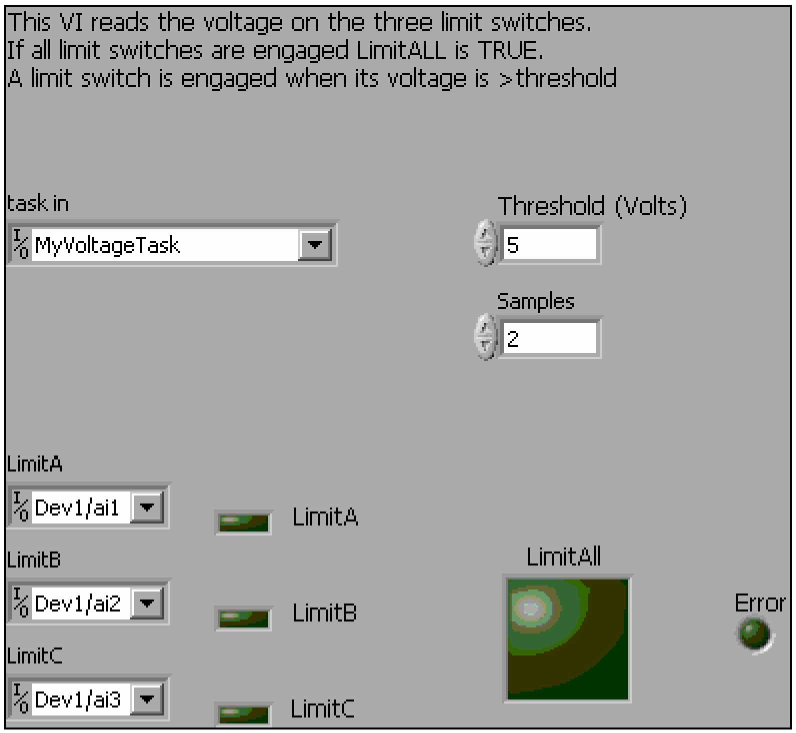


LimitAll.vi

Connector Pane



Front Panel



Controls and Indicators

I/O **task in task in** specifies the task to which to add the virtual channels this VI creates. If you do not specify a task, NI-DAQmx creates a task for you and adds the virtual channels this VI creates to that task.

LimitA physical channels specifies the names of the physical channels to use to create virtual channels. The DAQmx Physical Channel Constant lists all physical channels on devices and modules installed in the system.

LimitB physical channels specifies the names of the physical channels to use to create virtual channels. The DAQmx Physical Channel Constant lists all physical channels on devices and modules installed in the system.

LimitC physical channels specifies the names of the physical channels to use to create virtual channels. The DAQmx Physical Channel Constant lists all physical channels on devices and modules installed in the system.

Threshold (Volts)

Samples

LimitA

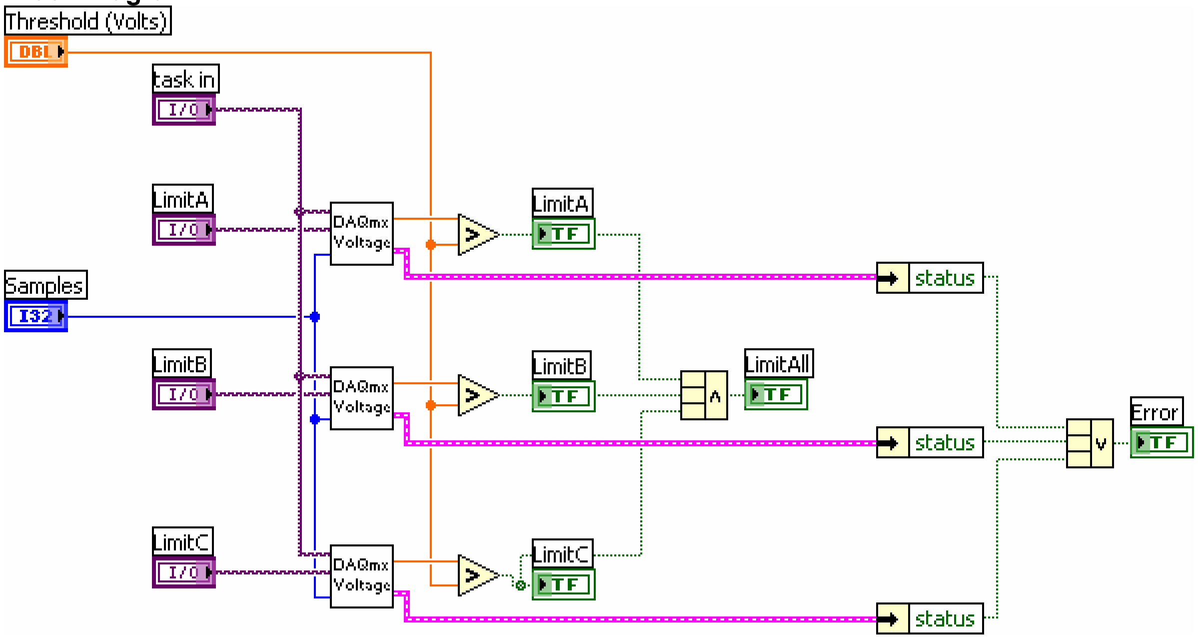
LimitB

LimitC

LimitAll

Error

Block Diagram

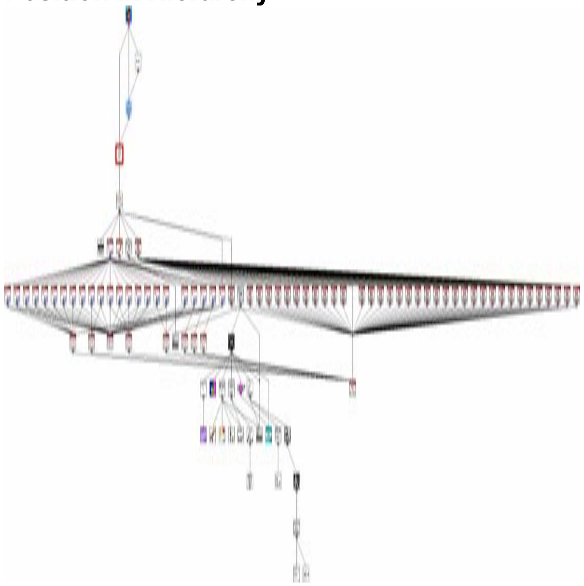


List of SubVIs and Express VIs with Configuration Information

DAQ_AI.vi
D:\Copy of Goniometer VI\DAQ_AI.vi

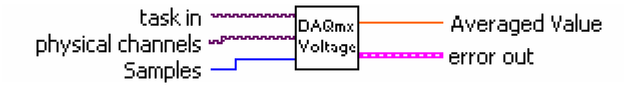
VI Revision History
"LimitAll.vi History"
Current Revision: 6

Position in Hierarchy

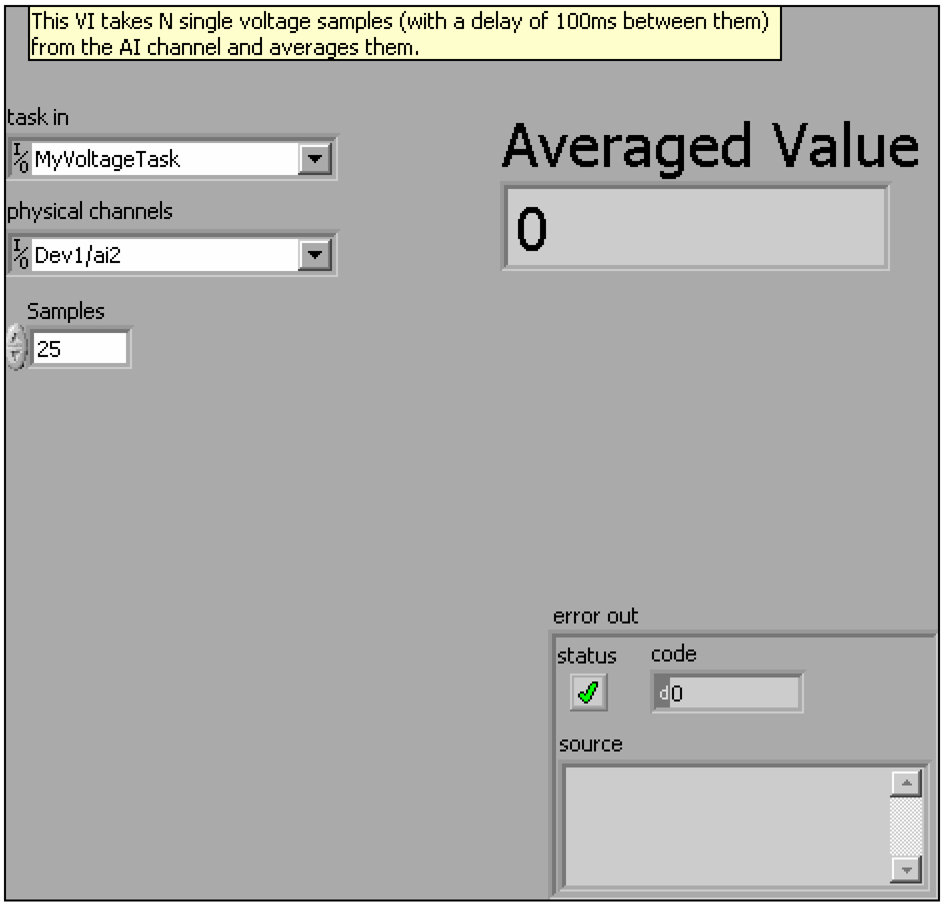


DAQ_AI.vi

Connector Pane



Front Panel



Controls and Indicators

I/O **task in task in** specifies the task to which to add the virtual channels this VI creates. If you do not specify a task, NI-DAQmx creates a task for you and adds the virtual channels this VI creates to that task.

I/O **physical channels physical channels** specifies the names of the physical channels to use to create virtual channels. The DAQmx Physical Channel Constant lists all physical channels on devices and modules installed in the system.

I32 **Samples**

error out error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces.

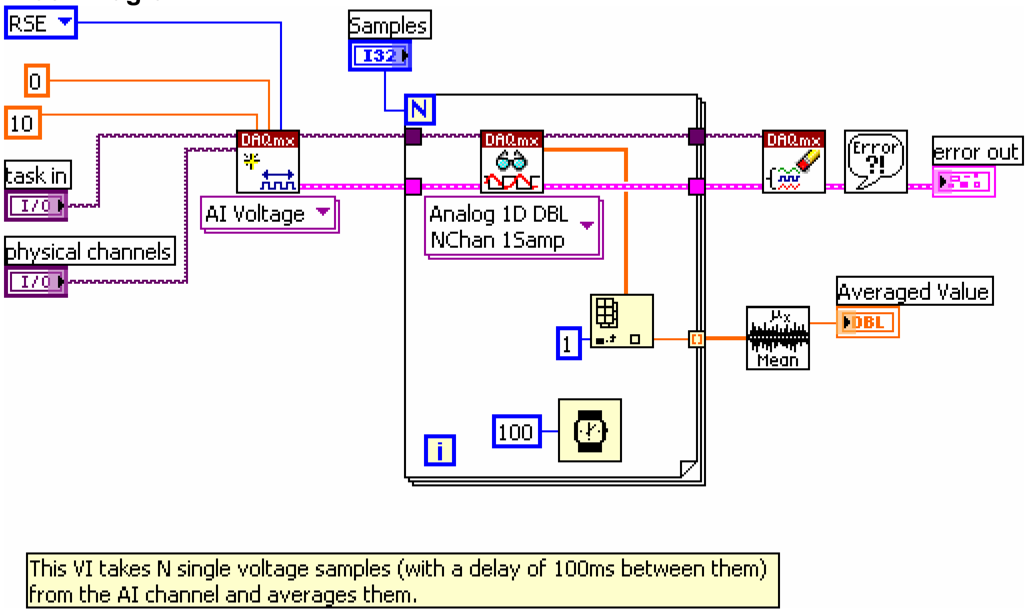
TF **status status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

I32 **code code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

abc **source source** identifies where and why an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

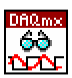
DBL **Averaged Value**

Block Diagram



List of SubVIs and Express VIs with Configuration Information

DAQmx Create Virtual Channel.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\DAQmx\create\channels.llb\DAQmx Create Virtual Channel.vi

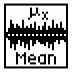


DAQmx Read.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\DAQmx\read.llb\DAQmx Read.vi

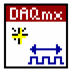
DAQmx Clear Task.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\DAQmx\configure\task.llb\DAQmx Clear Task.vi

Simple Error Handler.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\Utility\error.llb\Simple Error Handler.vi

DAQmx Read (Analog 1D DBL NChan 1Samp).vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\DAQmx\read.llb\DAQmx Read (Analog 1D DBL NChan 1Samp).vi



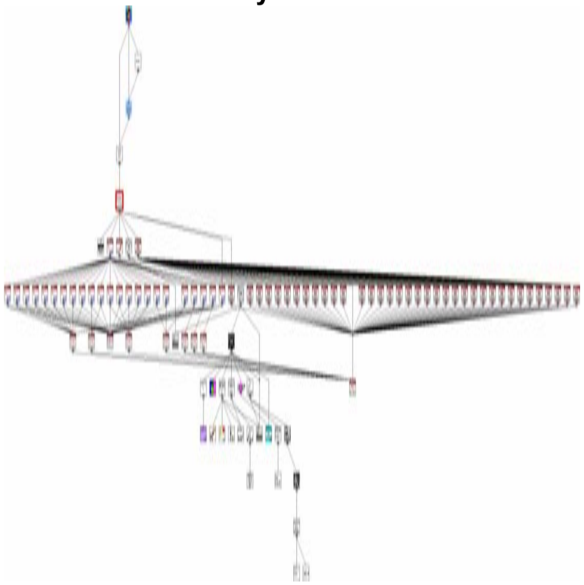
Mean.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\Analysis\baseanly.llb\Mean.vi



DAQmx Create Channel (AI-Voltage-Basic).vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\DAQmx\create\channels.llb\DAQmx Create Channel (AI-Voltage-Basic).vi

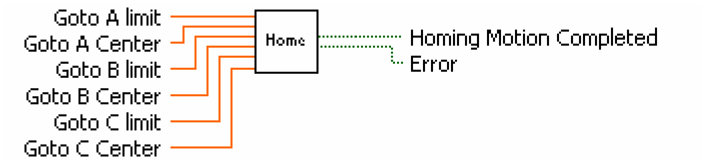
VI Revision History
"DAQ_AI.vi History"
Current Revision: 13

Position in Hierarchy

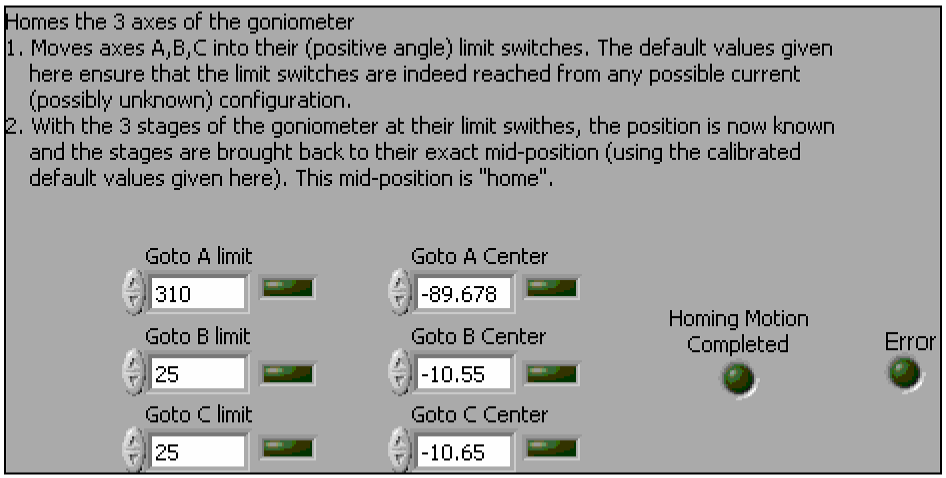


Home.vi

Connector Pane



Front Panel



Controls and Indicators

DBL **Goto A limit**

DBL **Goto B limit**

DBL **Goto C limit**

DBL **Goto A Center**

DBL **Goto B Center**

DBL **Goto C Center**

TF **Boolean**

TF **Boolean 2**

TF **Boolean 3**

TF **Boolean 4**

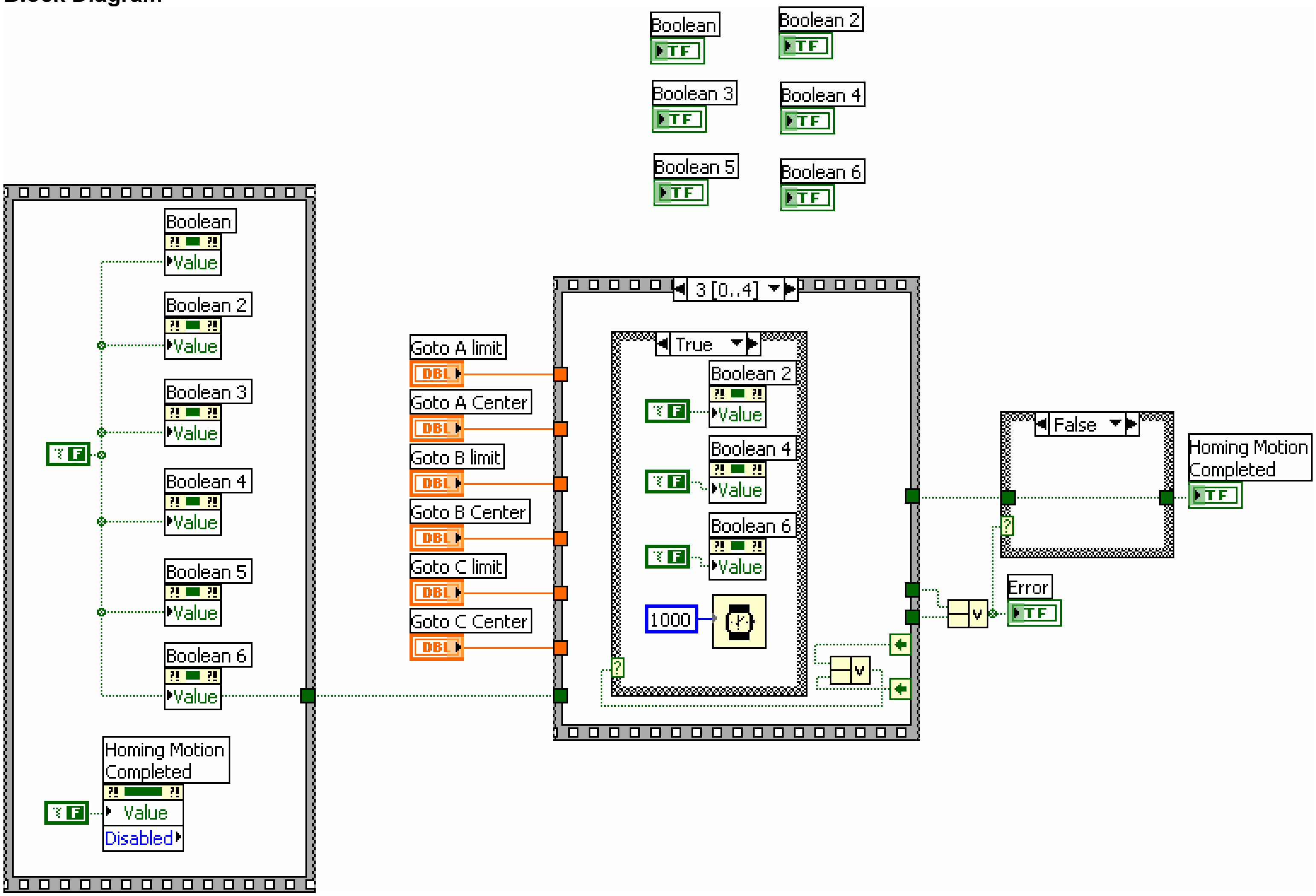
TF **Boolean 5**

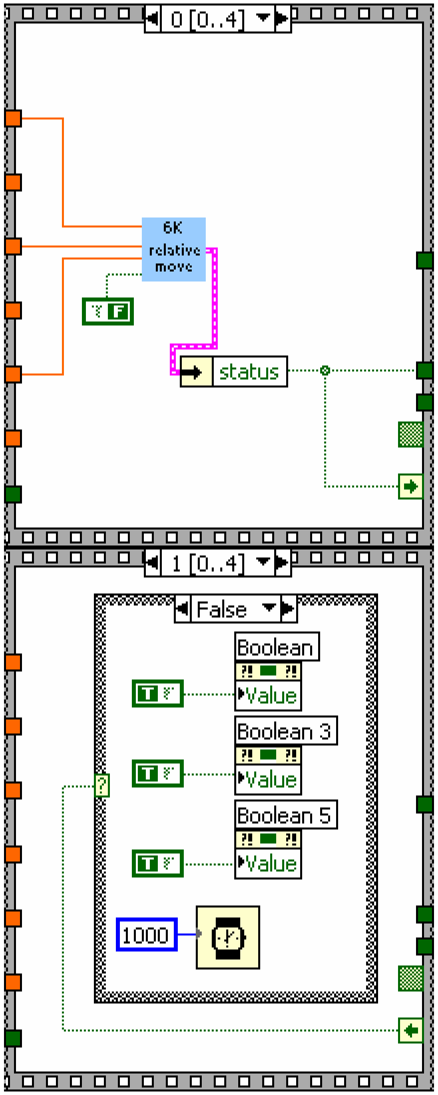
TF **Boolean 6**

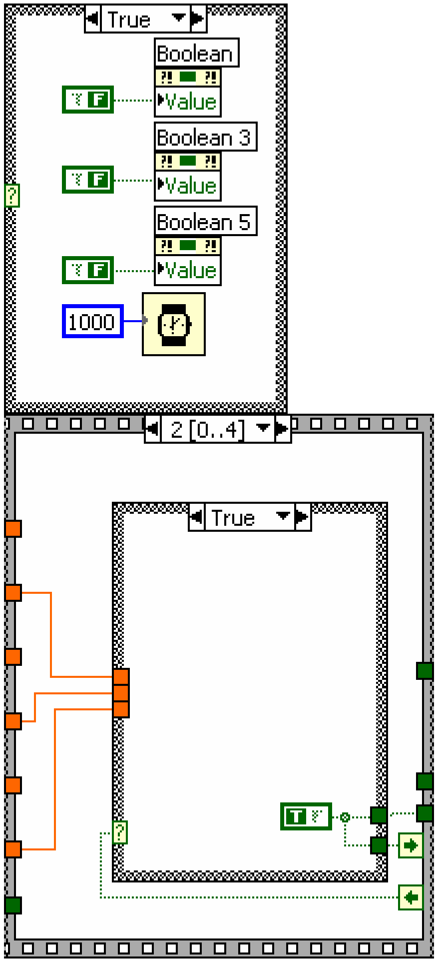
TF **Homing Motion Completed**

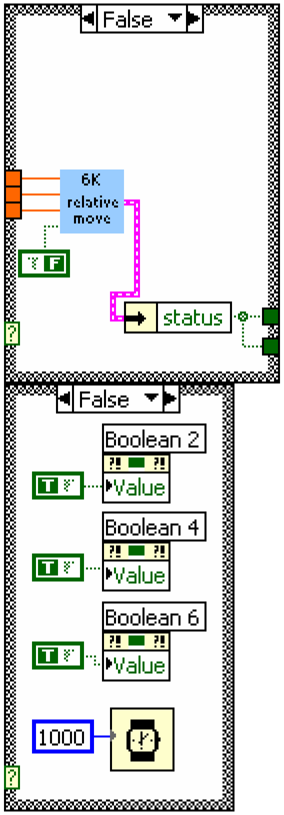
TF **Error**

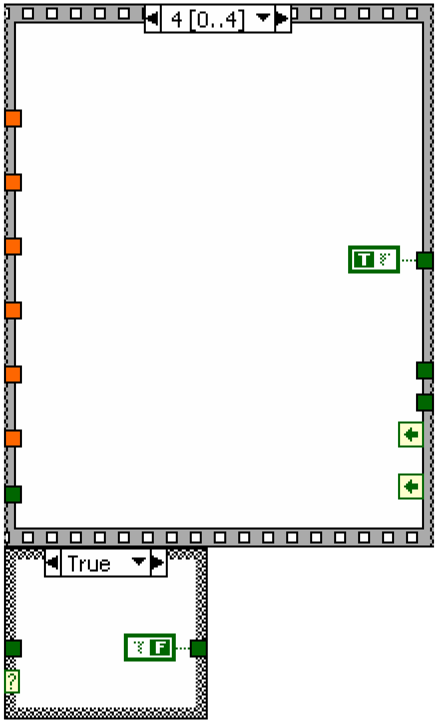
Block Diagram




0 [0..4]
6K relative move
status
1 [0..4]
False
Boolean
Value
Boolean 3
Value
Boolean 5
Value
1000


True
Boolean
Value
Boolean 3
Value
Boolean 5
Value
1000
2 [0..4]
True


False
6K relative move
status
False
Boolean 2
Value
Boolean 4
Value
Boolean 6
Value
1000

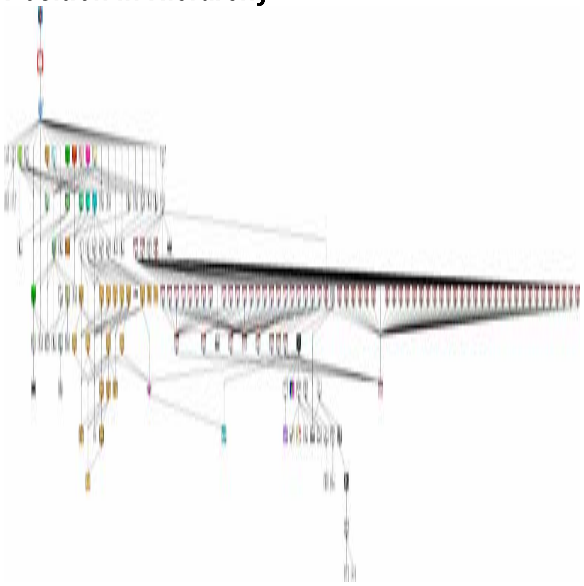


List of SubVIs and Express VIs with Configuration Information

6K Simple Move Basic_V3.vi
D:\Copy of Goniometer VI\6K Simple Move Basic_V3.vi

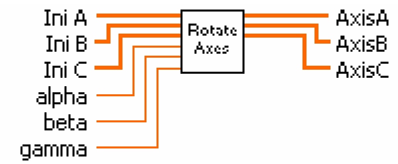
VI Revision History
"Home.vi History"
Current Revision: 27

Position in Hierarchy

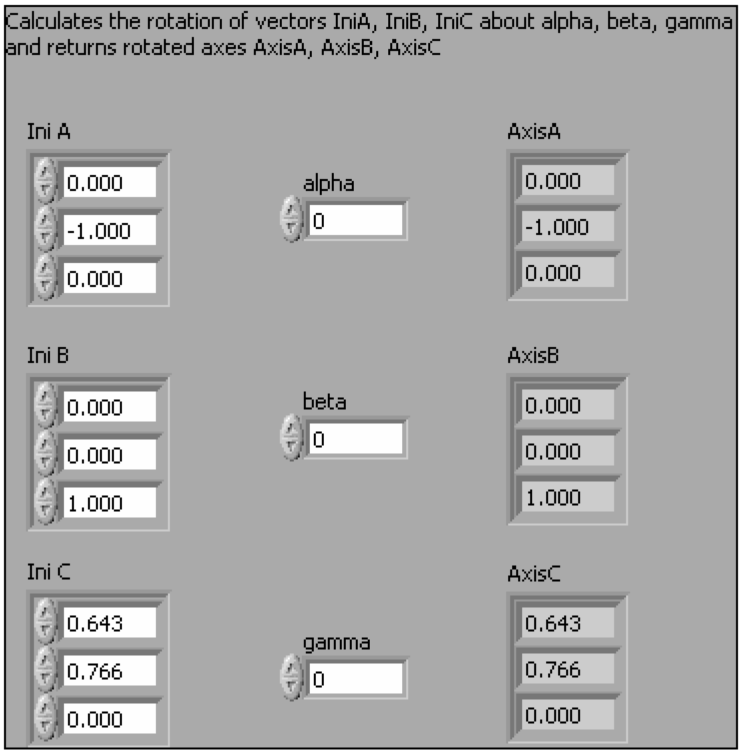


RotateAxes.vi

Connector Pane



Front Panel



Controls and Indicators

DBL **Ini A**

DBL **Numeric**

DBL **Ini B**

DBL **Numeric**

DBL **Ini C**

DBL **Numeric**

DBL **alpha**

DBL **beta**

DBL **gamma**

DBL **AxisA**

DBL **Numeric**

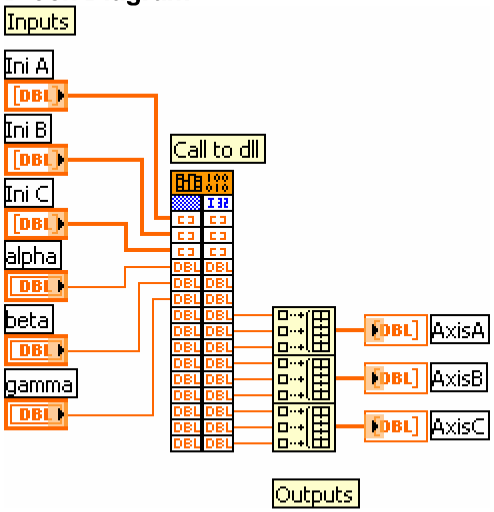
DBL **AxisB**

DBL **Numeric**

DBL **AxisC**

DBL **Numeric**

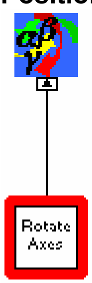
Block Diagram



List of SubVIs and Express VIs with Configuration Information

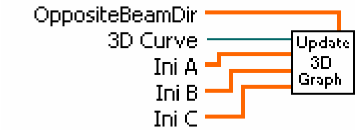
VI Revision History
"RotateAxes.vi History"
Current Revision: 9

Position in Hierarchy

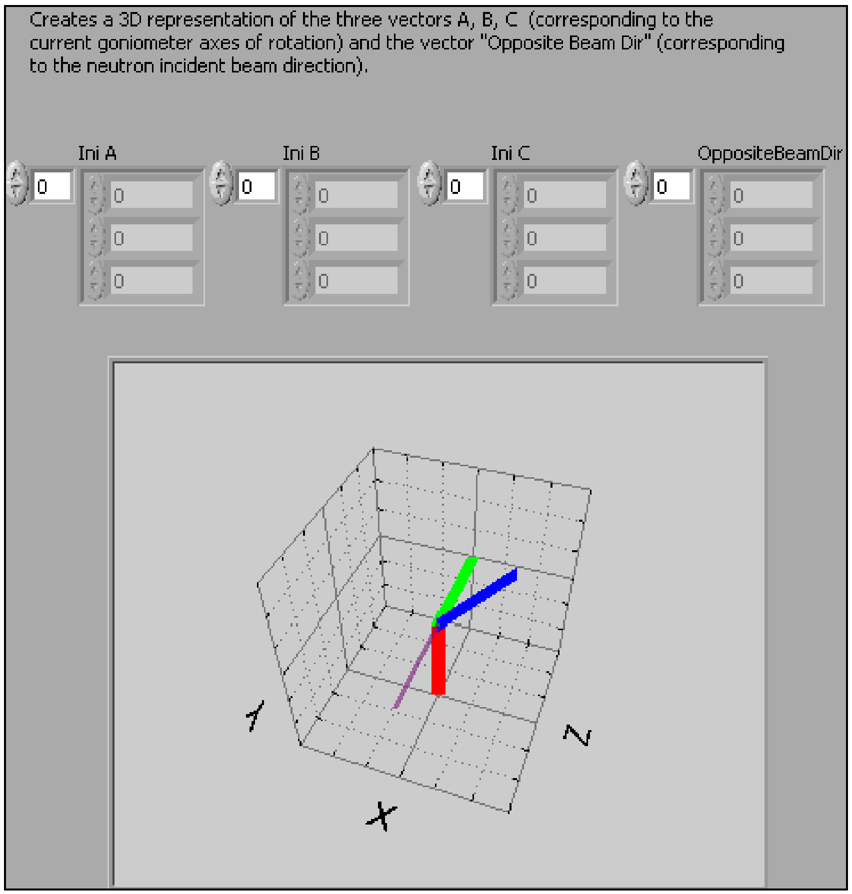


3D_Graph.vi

Connector Pane



Front Panel



Controls and Indicators

3D Curve ActiveX container that holds the 3D graph control.

Ini C

Numeric

Ini A

Numeric

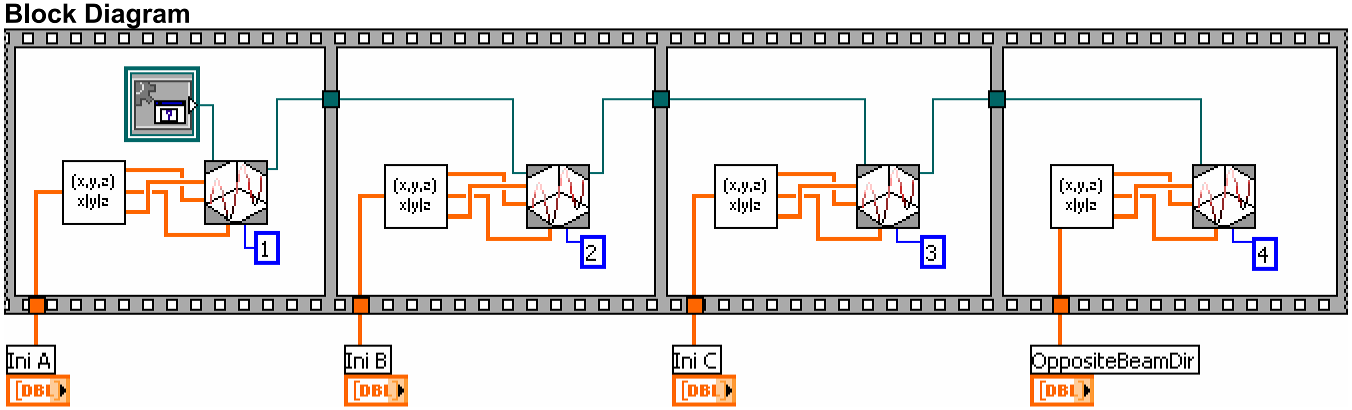
Ini B

Numeric

OppositeBeamDir

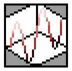
Numeric

Block Diagram



List of SubVIs and Express VIs with Configuration Information

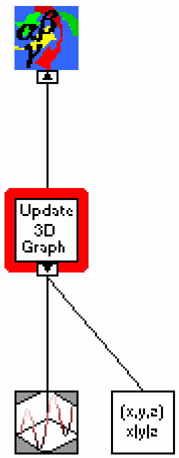
VectorTransform.vi
D:\Copy of Goniometer VI\VectorTransform.vi



3D Curve.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\Platform\3dgraph.llb\3D Curve.vi

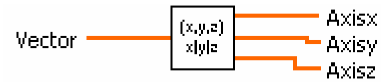
VI Revision History
"3D_Graph.vi History"
Current Revision: 2

Position in Hierarchy

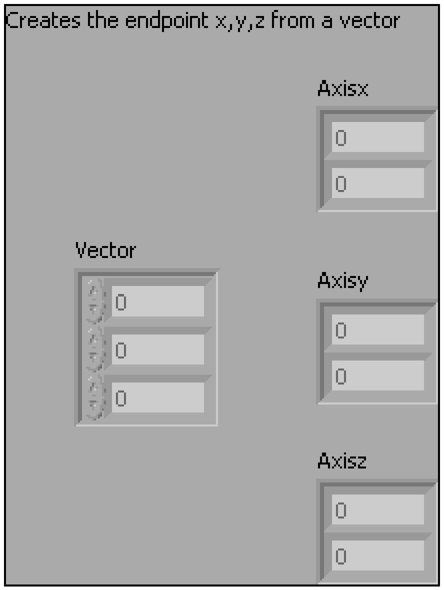


VectorTransform.vi

Connector Pane



Front Panel



Controls and Indicators

Vector

- **Numeric**

Axisx

- **Numeric**

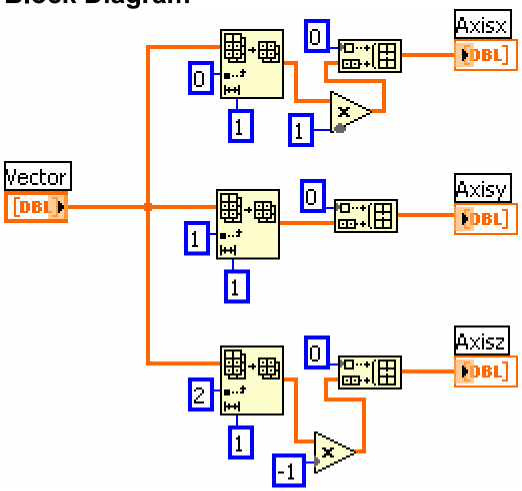
Axisy

Numeric

Axisz

Numeric

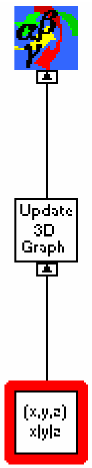
Block Diagram



List of SubVIs and Express VIs with Configuration Information

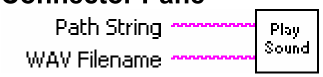
VI Revision History
"VectorTransform.vi History"
Current Revision: 7

Position in Hierarchy

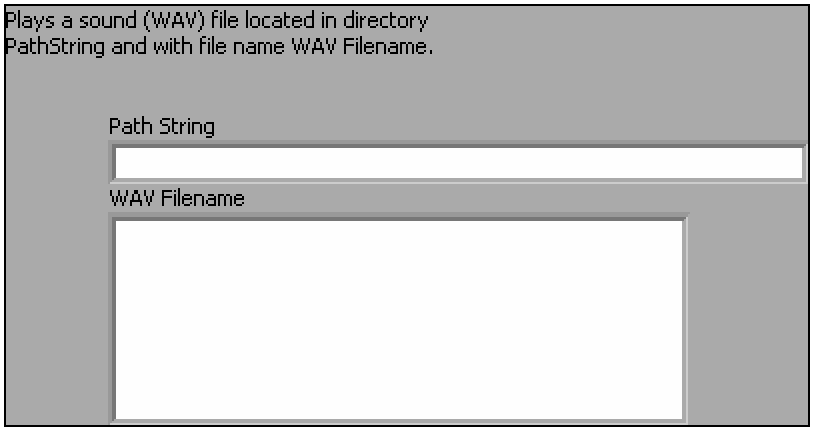


PlaySound.vi

Connector Pane



Front Panel

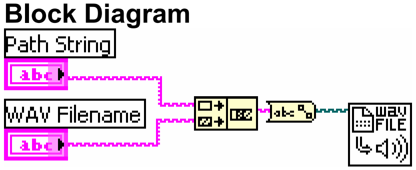


Controls and Indicators

Path String

WAV Filename

Block Diagram



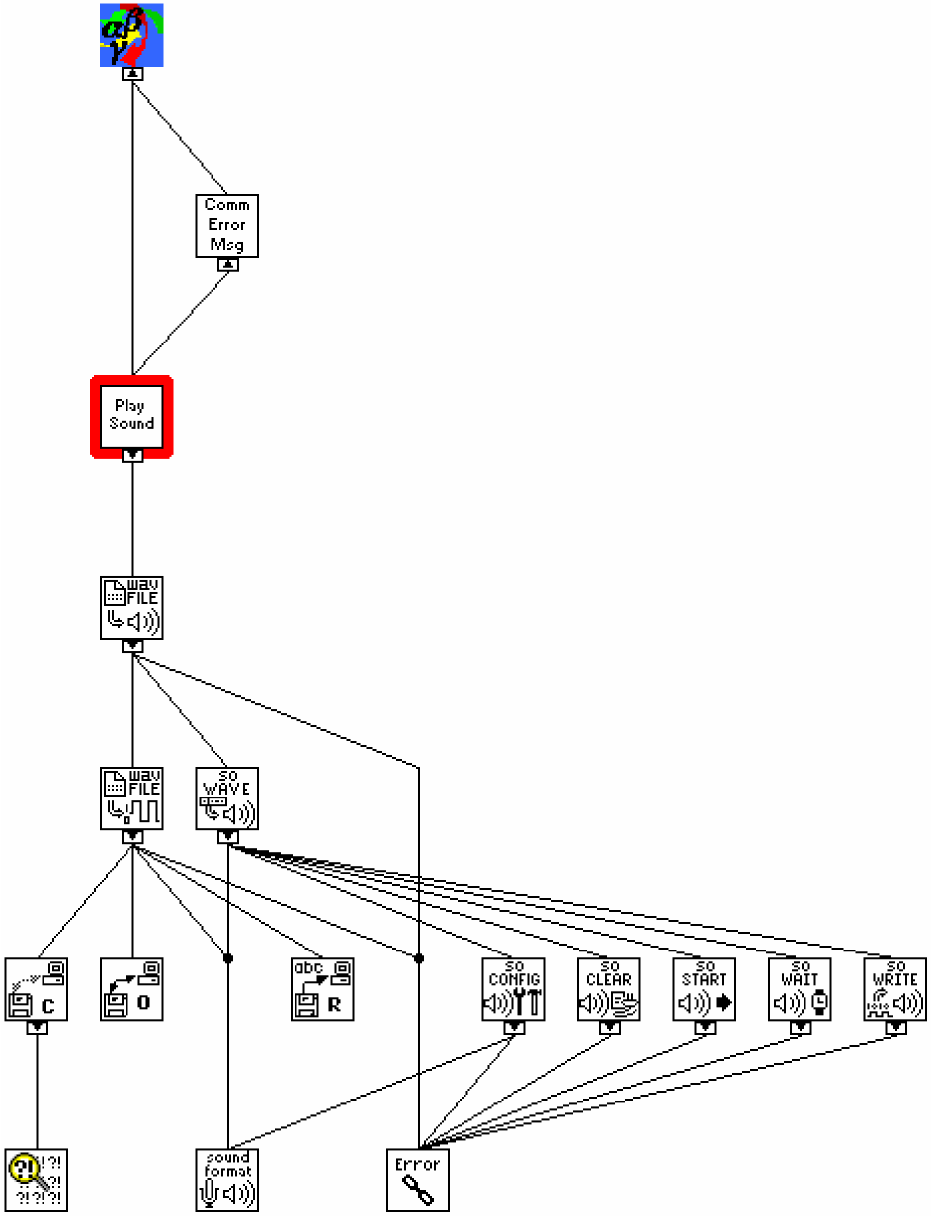
List of SubVIs and Express VIs with Configuration Information

Snd Play Wave File.vi
C:\Program Files\National Instruments\LabVIEW 7.0\vi.lib\sound\lvsound.llb\Snd Play

Wave File.vi

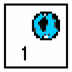
VI Revision History
"PlaySound.vi History"
Current Revision: 4

Position in Hierarchy



Globals.vi

Connector Pane



Front Panel

Controls and Indicators

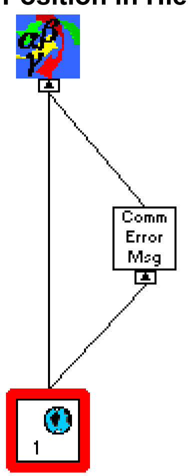
Path String

List of SubVIs and Express VIs with Configuration Information

VI Revision History

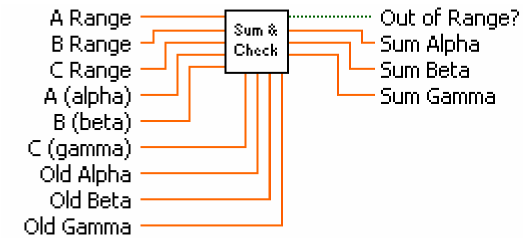
"Globals.vi History"
Current Revision: 1

Position in Hierarchy

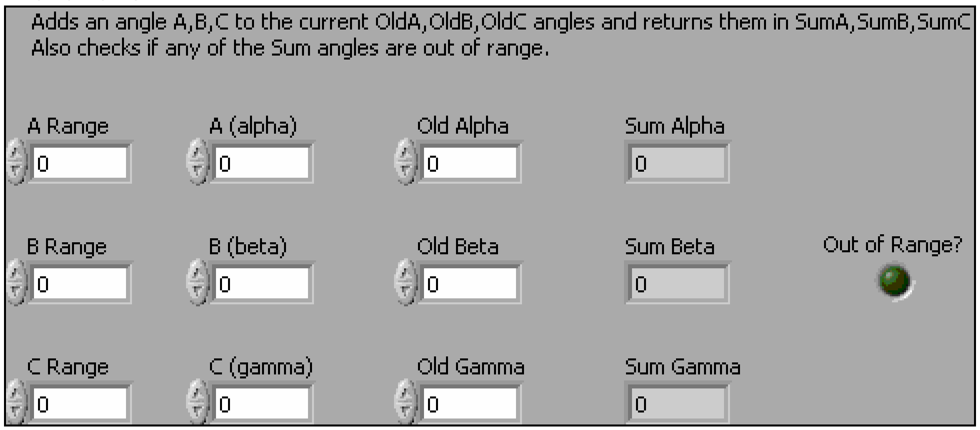


SumAndRangeCheck.vi

Connector Pane



Front Panel



Controls and Indicators

DBL **Old Gamma**

DBL **Old Beta**

DBL **Old Alpha**

DBL **A (alpha)**

DBL **B (beta)**

DBL **C (gamma)**

DBL **C Range**

DBL **B Range**

DBL **A Range**

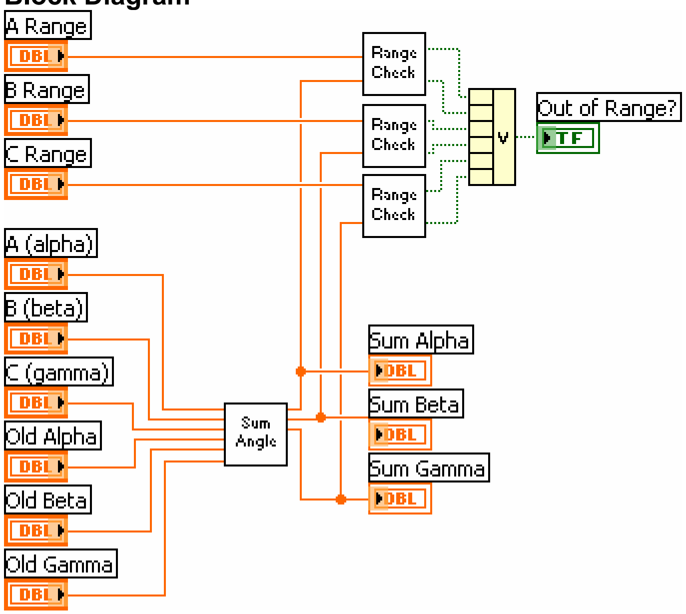
DBL **Sum Gamma**

DBL **Sum Beta**

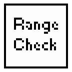
DBL **Sum Alpha**

TF **Out of Range?**

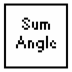
Block Diagram



List of SubVIs and Express VIs with Configuration Information



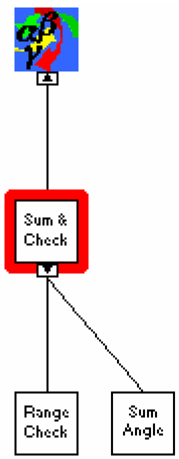
RangeCheck.vi
D:\Copy of Goniometer VI\RangeCheck.vi



SumAngles.vi
D:\Copy of Goniometer VI\SumAngles.vi

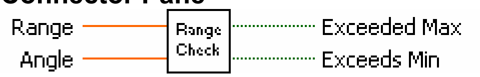
VI Revision History
"SumAndRangeCheck.vi History"
Current Revision: 5

Position in Hierarchy

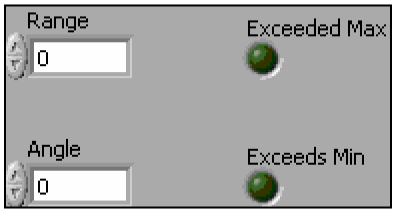


RangeCheck.vi

Connector Pane



Front Panel



Controls and Indicators

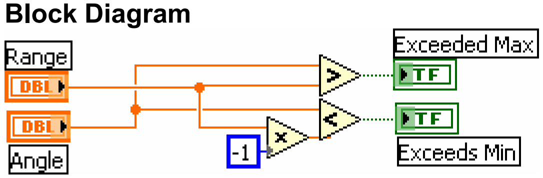
DBL **Angle**

DBL **Range**

TF **Exceeds Min**

TF **Exceeded Max**

Block Diagram

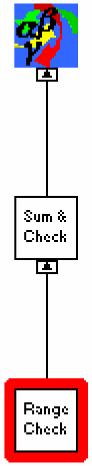


List of SubVIs and Express VIs with Configuration Information

VI Revision History

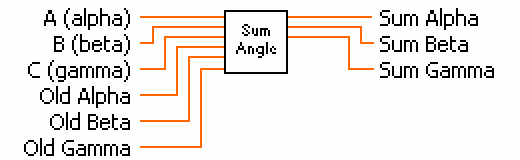
"RangeCheck.vi History"
Current Revision: 2

Position in Hierarchy

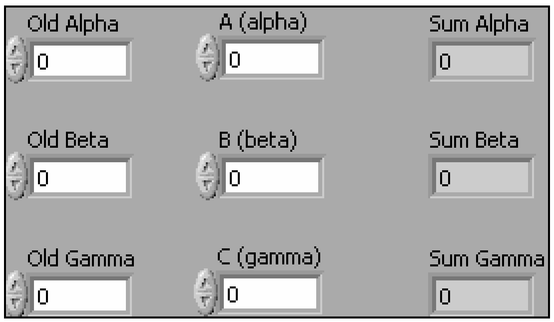


SumAngles.vi

Connector Pane



Front Panel



Controls and Indicators

DBL **Old Gamma**

DBL **Old Beta**

DBL **Old Alpha**

DBL **C (gamma)**

DBL **B (beta)**

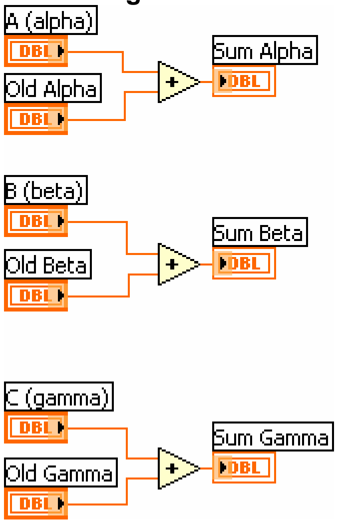
DBL **A (alpha)**

DBL **Sum Gamma**

DBL **Sum Beta**

DBL **Sum Alpha**

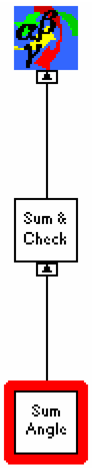
Block Diagram



List of SubVIs and Express VIs with Configuration Information

VI Revision History
"SumAngles.vi History"
Current Revision: 2

Position in Hierarchy



CurrentDirectoryString.vi

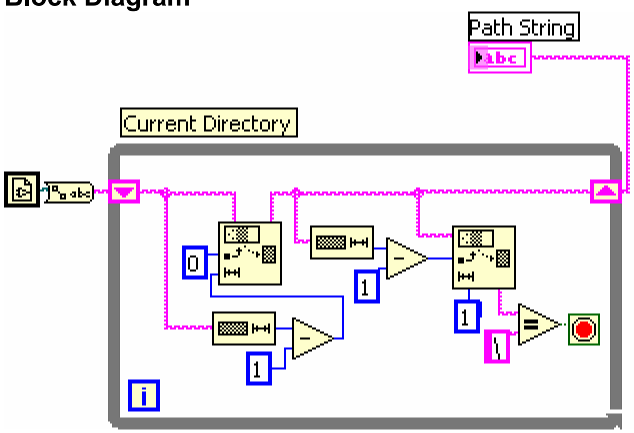
Connector Pane

Front Panel

Controls and Indicators

 **Path String**

Block Diagram



List of SubVIs and Express VIs with Configuration Information

VI Revision History

"CurrentDirectoryString.vi History"

Current Revision: 2

Position in Hierarchy